Leonardo Gabriel Sousa Coelho

# Plataforma web para gestão de processos ETL em ambientes multi-instituição.

# Web platform for ETL process management in multi-institution environments.

Leonardo Gabriel Sousa Coelho

# Plataforma web para gestão de processos ETL em ambientes multi-instituição.

# Web platform for ETL process management in multi-institution environments.

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Associado com Agregação José Luís Oliveira, Professor do Departamento Eletrónica e Telecomunicações e Informática da Universidade de Aveiro.

**o júri / the jury**

presidente / president

**Prof. Dr. Augusto Marques Ferreira da Silva**
Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro)

vogais / examiners committee

**Prof. Dr. José Luís Oliveira**
Professor Associado com Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

**Prof. Dr. Rui Pedro Sanches de Castro Lopes**
Professor Coordenador do Departamento de Informática e Comunicações do Instituto Politécnico de Bragança

**agradecimentos / acknowledgements**

Inicialmente, quero agradecer ao meu orientador (Professor José Luís Oliveira) pelo apoio, interesse e simpatia demonstrados ao longo de todo o trabalho. O seu sentido crítico foi muito importante para a qualidade do trabalho desenvolvido e para o meu crescimento pessoal.

Em segundo lugar, quero agradecer ao meu co-orientador (Renato Pinho) e aos restantes colaboradores da BMD Software (Luís Bastião e Eriksson Monteiro). A vossa disponibilidade e colaboração foram inigualáveis. Sinto que este foi o ano que mais evoluí técnicamente, sobretudo graças a vocês.

Agradeço ainda a oportunidade proporcionada de fazer parte integrante do grupo de bioinformática do IEETA da Universidade de Aveiro e do espiríto de entre-ajuda que os membros do grupo sempre manisfestaram.

Por fim, queria fazer o agradecimento mais importante. Agradeço à minha família por todo o apoio que me ofereceram ao longo de todo este percurso. Permitindo a estabilidade emocional que necessitava para ter sucesso naquilo que mais gosto.

**Resumo**

A evolução tecnológica e a digitização sucessiva de processos e serviços têm criado novas oportunidades no domínio da integração de dados. A promessa de tomada de decisões baseada em estatísticas tem sido amplamente reconhecida. Nesse sentido, Data Warehouses são estruturas que procuram armazenar todas as informações relevantes de um negócio em particular, possibilitando a análise eficiente de grandes volumes de dados, oferecendo suporte à tomada de decisões e à previsão de eventos futuros. Uma Data Warehouse pode integrar dados de múltiplas fontes. Sendo assim, os dados presentes nas diversas fontes de dados heterogéneas devem ser devidamente capturados, tratados e uniformizados. Os processos ETL respondem a essa necessidade, permitindo a definição de um fluxo de trabalho programado que combina funções de extração, transformação e carregamento de dados.

O entusiasmo por ferramentas que permitem o desenvolvimento de processos ETL de uma forma visual tem vindo a aumentar, pois tornam o processo mais simples e intuitivo. Contudo, atualmente não existem ferramentas web robustas e completas para o desenvolvimento de processos ETL. Uma solução web permite a instalação centralizada da aplicação, libertando o desenvolvedor dos processos ETL da necessidade de gerir a instalação da aplicação, assim como as suas dependências e conectores de bases de dados.

Esta dissertação teve como principal objetivo o desenvolvimento de uma solução web completa e robusta que permitisse o desenvolvimento e gestão de processos ETL num contexto multi-instituição. Sabendo que os dados usados são tipicamente sensíveis, era necessário garantir a privacidade e proteção dos mesmos. Além disso, os processos ETL são executados periodicamente para atualizar a Data Warehouse ou para a produção de relatórios estatísticos, sendo necessária a capacidade de execução escalonada e periódica dos processos.

**Abstract**

The technological evolution and the successive digitization of processes and services created new opportunities for data collection in a wide range of application areas. The promise of statistics-driven decision-making is now being widely recognized. Data Warehouses seek to store all relevant information for a particular business, allowing the efficient analysis of large volumes of data, supporting decision-making and future events prediction. A Data Warehouse may integrate data from multiple data sources. Hence, the data present in the various heterogeneous data sources must be properly captured, parsed and standardized. ETL processes adress this need, allowing the definition of a programming workflow which combines data extraction, transformation and loading functions.

The enthusiasm for tools that allow the development of ETL processes in a visual way has been increasing, because they make the process simpler and more intuitive. However, there are currently no robust and comprehensive web tools for developing ETL processes. A web solution enables centralized application installation, freeing the ETL process developer from the need to manage the application installation, as well as its dependencies and database connectors.

The main goal of this dissertation was to develop a robust and comprehensive web solution for the development and management of ETL processes in a multi-institution environment. Knowing that the used data is typically sensitive, it was necessary to ensure data privacy and protection. In addition, ETL processes are periodically executed to update the Data Warehouse or to produce statistical reports, requiring the ability to define scheduled and periodic execution of ETL processes.

# Contents

**Contents** i

**List of Figures** iii

**List of Tables** v

**Acronyms** 1

**1 Introduction** 3
- 1.1 Motivation and Goals . . . 3
- 1.2 Thesis outline . . . 5

**2 Background & State of the Art** 7
- 2.1 Big Data . . . 7
  - 2.1.1 High-level pipeline . . . 7
- 2.2 Business Intelligence . . . 9
- 2.3 ETL tools . . . 11
  - 2.3.1 Open-Source Tools . . . 12
  - 2.3.2 Open-Source tools vs Commercial tools . . . 15
- 2.4 Business Analytics Tools . . . 15
  - 2.4.1 Power BI . . . 16
  - 2.4.2 Metabase . . . 16
  - 2.4.3 Saiku . . . 18
- 2.5 Business Intelligence Suites . . . 18
  - 2.5.1 Pentaho . . . 19
  - 2.5.2 SpagoBI . . . 20
  - 2.5.3 Jaspersoft . . . 21
  - 2.5.4 BI Suites comparison . . . 22
- 2.6 Summary . . . 23

**3 Proposal Definition** 25
- 3.1 Functional Requirements . . . 25
- 3.2 Non Functional Requirements . . . 27
- 3.3 System Modeling . . . 27
  - 3.3.1 Functional Model . . . 27
  - 3.3.2 Data Model . . . 28
  - 3.3.3 Overall Architecture . . . 32

3.4 Summary . . . . . 35

**4 System Implementation** **37**
4.1 The pipeline editor . . . . . 37
4.2 ETL SDK . . . . . 37
4.2.1 Pentaho Data Integration . . . . . 37
4.2.2 DI SDK . . . . . 39
4.2.3 Rendering algorithm . . . . . 42
4.2.4 Extensibility . . . . . 44
4.3 Execution servers . . . . . 44
4.4 Task scheduler . . . . . 46
4.5 Summary . . . . . 47

**5 BICenter: Results and Use cases** **49**
5.1 BIcenter User Interface . . . . . 49
5.2 A use case on biomarkers analysis . . . . . 55

**6 Conclusion** **57**
6.1 Future Work . . . . . 57

**Bibliography** **59**

# List of Figures


1.1 PDI components by category. . . . . 5

2.1 The Big Data Analysis Pipeline (adapted from [1]). . . . . 8
2.2 Business Intelligence Architecture. . . . . 10
2.3 Business Intelligence Pipeline. . . . . 10
2.4 A Dimensional Model of a Data Warehouse. . . . . 11
2.5 Overview of Kettle programs. Source by [2] . . . . . 13
2.6 Power BI ecosystem. . . . . 17
2.7 Metabase dashboard. . . . . 17
2.8 Saiku dashboard. . . . . 18
2.9 Pentaho Business Intelligence Suite. . . . . 19
2.10 Pentaho Architecture. . . . . 20
2.11 SpagoBI Architecture. . . . . 21
2.12 Jaspersoft Architecture. . . . . 22

3.1 System use cases. . . . . 26
3.2 System workfow to build, manage and execute ETL tasks. . . . . 28
3.3 Example of an ETL process. . . . . 29
3.4 ETL process ER model. . . . . 29
3.5 Institution ER model. . . . . 30
3.6 Execution ER model. . . . . 31
3.7 RBAC ER model. . . . . 32
3.8 Components diagram. . . . . 33
3.9 Deployment diagram. . . . . 34
3.10 MVC architecture. . . . . 34

4.1 MxGraph architecture. . . . . 38
4.2 ETL task execution flowchart. . . . . 39
4.3 ETL task construction flowchart. . . . . 39
4.4 Trans builder class diagram. . . . . 41
4.5 TransExecutor class. . . . . 41
4.6 Boolean condition . . . . . 43
4.7 SortRows component specification. . . . . 44
4.8 Slave server configuration. . . . . 45
4.9 Dynamic Carte cluster configuration . . . . . 46

5.1 Login page. . . . . 49

5.2 Main page. . . . . 50
5.3 Editor toolbar. . . . . 50
5.4 Institution page. . . . . 50
5.5 Count rainy days by month. . . . . 51
5.6 Resource settings . . . . . 51
5.7 Filter rows settings page. . . . . 52
5.8 Filter rows input fields. . . . . 52
5.9 Scheduling a ETL task on a remote execution server. . . . . 53
5.10 Execution scheduling table. . . . . 53
5.11 Execution history. . . . . 54
5.12 Execution logs. . . . . 54
5.13 Execution step measures. . . . . 54
5.14 Preview execution output. . . . . 55
5.15 ETL task to calculate biomarkers averages. . . . . 55
5.16 Biomarkers averages. . . . . 56

# List of Tables

2.1 Criteria fulfillment by the candidates . . . . . 14

3.1 List of user stories . . . . . 26
3.2 List of supported ETL components . . . . . 27
3.3 Data source fields . . . . . 30
3.4 Execution server fields . . . . . 30
3.5 Step Metrics fields . . . . . 31

4.1 Cron expression fields. . . . . 46

# Acronyms

- **AD:** Active Directory
- **API:** Application Programing Interface
- **BA:** Business Analytics
- **BI:** Business Intelligence
- **DI:** Data Integration
- **ETL:** Extract, Transform and Load
- **ER:** Entity-Relationship
- **GWT:** Google Web Toolkit
- **IPC:** Informatica Power Center
- **LDAP:** Lightweight Directory Access Protocol
- **MVC:** Model-View-Controller
- **OLAP:** Online Analytical Processing
- **PDI:** Pentaho Data Integration
- **RBAC:** Role Based Access Control
- **SDK:** Software Development Kit
- **TOS:** Talend Open Studio

# Chapter 1

# Introduction

## 1.1 Motivation and Goals

The technological evolution and the successive digitization of processes and services created new opportunities for data collection in a wide range of application areas. The promise of statistics-driven decision-making is now being widely recognized, mainly due to a developing enthusiasm around the Big Data concept. However, currently there is a discrepancy between the potential of Big Data and its realization. Heterogeneity, scale, timeliness, complexity and privacy problems, limit the progress at all phases of the pipeline that can create value from the data [3, 4, 5, 6].

Business Intelligence (BI) allows the implementation of processes to analyze and present Big Data in a useful way for a particular business. BI includes a wide range of tools, applications and methodologies which empowers organizations with internal and external data collection capabilities, which allows data preparation for analysis, and enables to develop and execute queries against the data, to provide visually appealing analytical results for decision makers, namely reports, dashboards and data visualizations [7, 8]. BI software can be represented by three broad application categories: data management, data discovery, and reporting tools. Data management tools typically belong to the bottom tier, where data needs to be collected, transformed and loaded to the target system, usually a data warehouse. Data discovery applications, related to the middle tier, provides the ability to sift through data and build business value conclusions. Reporting tools, in front tier, are keen on the translation of business value conclusions into more user-friendly information, such as graphs or customized reports.

Data Warehousing simplify the capture and analysis of data from multiple sources. However, the process of data integration and transformation into a data warehouse typically consumes up to 80 percent of development resources [9]. In order to store data in an efficient way, the collected raw data must be properly parsed and handled. Extract, Transform and Load (ETL) is a programming workflow that addresses this need, through the combination of three distinct functions: 1) The Extract function reads and collects a useful subset, from distinct and heterogeneous data sources; 2) The Transform function manipulates the data to convert it to the desired format and state; 3) The Load function stores the resulting dataset into the target database. ETL tools are the key components in the construction of data warehouses, by managing all the input processes [10].

A typical ETL process consist in a pipeline that encompasses a finite number of transfor-

mation steps applied over a data stream. Considering a real-life complex scenario, the number of transformation steps will grow exponentially. Due to the pipeline complexity growth, the pipeline management will become unbearable and highly susceptible to errors. Thus, it is crucial to build tools that facilitates the management of ETL processes.

Various GUI-based tools have been built aiming to simplify the construction of ETL processes. To help understanding and identifying the most relevant and complete graphical ETL tools available on the market, several surveys have been presented [11, 12]. Briefly, both Talend Open Studio (TOS) and Pentaho Data Integration (PDI or Kettle) are recognized as the most powerful open-source tools currently available, despite the increasing relevance of new tools such as CloverETL. In these tools the user is able to define a ETL process by building a visual pipeline formed by ETL components. The pipeline is built using drag-and-drop actions and ETL components configuration.

One of the main focus of these GUI-based tools is to support a higher level of engagement of users without a deep technical expertise. Nonetheless, current ETL tools, namely PDI, TOS and CloverETL, are all desktop-based, a characteristic that implies some technical skills, namely the ability to install the solution, the platform dependencies and the database connectors. Since ETL GUI-based tools seek to approach non-IT people, this problem appears as a contradiction to the tools' original purpose.

The aforementioned problems have been already identified by other authors, who also argued that a web-based solution would have a positive impact on the creation of a more transparent environment for data integration, facilitating the creation of ETL pipelines to non-IT people. In a web solution, the installation would be centralized, and the system manager would be in charge of all underlying details.

Krishna et al. [13] firstly proposed a web-based framework for representing data extraction from one or more data sources, transformation business logics and the load of data within a data warehouse. Later [14], they provide the capability to pre-configure multi-source connections that can be used in future transformations. The application also offers a viewable transformation report with execution metrics, in order to measure data quality and accuracy. In a distinct work, Novak et al. [15] proposed a prototype of a web ETL tool. The authors emphasize that as far as they knew there was no such thing as a complete and fully web-based open-source ETL tool. They also stated that with a web-based solution users could save time and space required for installation purposes, the application would be easily accessible anywhere via web browser and multiple users could work simultaneously.

However, existing web ETL solutions are quite limited, since they only encompass a very restricted set of ETL components. On the other hand, the existent desktop-based ETL tools (such as PDI or TOS) offers an endless number of components. For instance, Pentaho Data Integration latest version (v8.0) provides 237 ETL components, divided by 23 different categories (Figure 1.1). Considering the data heterogeneity of Big Data environments, it is interesting to know that PDI supports an enormous amount of different types of data sources, namely various types of structured, unstructured and distributed databases, OLAPs, text files, mails [1].

Typically, desktop-based ETL tools use an application programing interface (API) as the back-engine to empower the tool. Open-source ETL projects, such as Pentaho, Talend and CloverETL, make their ETL APIs public available. Hence, there is the possibility to use an already existent ETL API embedded in a web solution. This idea was already explored in

[1] https://wiki.pentaho.com

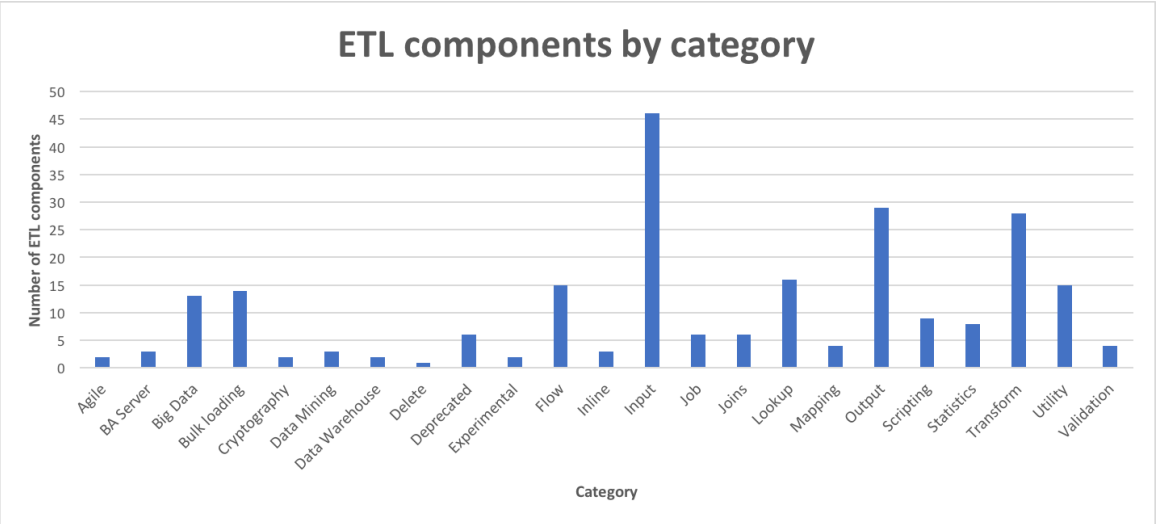



Figure 1.1: PDI components by category.

a project called SpoonWeb [2], a web-based graphical ETL designer tool that uses Kettle, the Pentaho's ETL API, as the platform's back-engine. SpoonWeb supports 125 of the 237 core ETL components that PDI offers (52,74%). Nonetheless, the effort of developing new ETL components is a very repetitive work.

The main goal of this dissertation is to build a complete and fully web-based platform that allows non-IT users to build and manage ETL processes within several institutions. In addition, knowing that the data is typically sensitive and should be restricted to the institution, the platform must ensures data privacy and protection. In general, ETL pipelines are executed periodically to update the data warehouse or to produce statistical reports. Thus, it is also important to allow scheduling periodic executions. Finally, considering the reproducibility of the integration of new ETL components, a strategy must be devised to facilitate their integration in the platform.

## 1.2 Thesis outline

The dissertation is organized in six chapters:

- **Chapter 1** outlines thesis motivation and goals. It explains the concept of ETL processes and stresses the importance of a complete and fully-functional web-based solution to manage them. Also, it addresses the challenges for managing the ETL processes in a multi-institution enviroment that has sensitive data.
- **Chapter 2** describes how Business Intelligence can be used to extract the potencial value of Big Data. Furthermore, it identifies and contextualizes the various tools that belongs to BI. A comparative analysis is also made between the several BI tools.
- **Chapter 3** presents the system requirements and how it was modelled. First there is an identification of all functional and non-functional requirements to develop the system

[2] https://github.com/HiromuHota/pentaho-kettle

and, from those requirements, it was made the proposal for implementation. It describes the proposed business process, the data model and the system overall architecture. Finally, it was explained the deployment strategy.

- **Chapter 4** explains the implementation decisions. Summarily, it performs a detailed description of the relevant aspects of each system logical component described in the overall architecture.
- **Chapter 5** presents the thesis results, by showing the system main workflow. It also presents a use case in the mass spectrometry scenario.
- **Chapter 6** presents the conclusion of this thesis and proposes future works.

# Chapter 2

# Background & State of the Art

## 2.1 Big Data

Big data is typically used to define large volumes of data, both structured or unstructured, that inundates a business on daily basis. Big Data environment are noisy and there will be a significant amount of invalid or corrupted records. Those records must be discarded during the acquisition phase. However, currently that decision is made in an ad hoc fashion. Furthermore, data are not natively in a structured format. Tweets and blogs are examples of weakly structured pieces of text. Images and videos are structured just for storage and display, but not for semantic content and search. Therefore, it is necessary to transform the content into a suitable structured format for later analysis. Nevertheless, this can be a huge challenge. Moreover, the value of data explodes when it is aggregated with other sources, however considering the data heterogeneity, data integration can be a major challenge. [16, 4, 5] Nevertheless, if these obstacles are overcome a huge improvement can be introduced to the business. Decisions that previously were made based on guesswork, or on artificial models of the reality, can now be made automatically using the data itself. [6]

### 2.1.1 High-level pipeline

The analysis of Big Data involves multiple distinct phases and each phase must be carefully thought in order to build a powerful solution. As the Figure 2.1 suggests, the main steps are:

1) This first step consists in collecting the data from several sources. Since part of the data is of no interest, this phase also considers the filtering and compression of the captured data. So, the definition of how data will be discarded, will be a major issue;

2) Seldom the collected information will be ready for analysis. So, the captured data cannot be left untouched, if an efficiently data analysis is expected. Rather, it is necessary to develop and apply information extraction processes to capture all relevant information from the underlying sources and translate it into the appropriate format for analysis. Performing this type of operations in a right and complete fashion represents a continuing technical challenge;

3) This phase considers the heterogeneity of the data flood, recognizing that it's not enough to merely record and stores it in a repository. Considering a non-hierarchical group of datasets in a given repository, it is unlikely that these data will be useful someday. However, the adequate meta-data will ease the usefulness of the data, still there will exist challenges to overcome. Data analysis is considerably more demanding than simply locating, identifying,

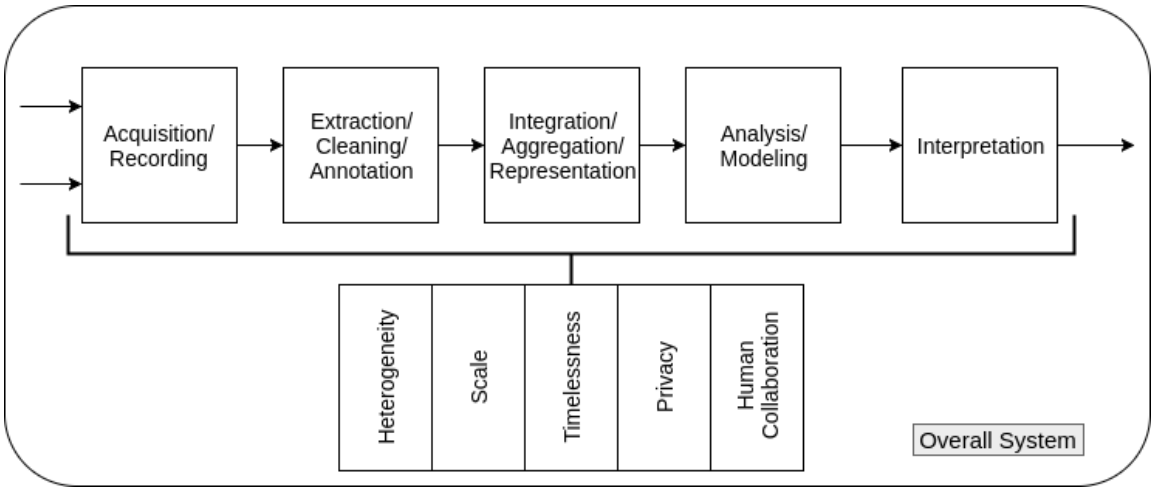



Figure 2.1: The Big Data Analysis Pipeline (adapted from [1]).

understanding, and citing data. For effective large-scale analysis all of this has to happen in a completely automated manner;

4) This is possibly the most important step of the analysis pipeline, where the existing data is actually processed and analyzed. Querying and mining methods for Big Data is substantially different from traditional statistical analysis on small samples. These data are often noisy, dynamic, heterogeneous, inter-related and untrustworthy. Nevertheless, even noisy, Big Data could be more valuable than tiny samples since general statistics obtained from frequent patterns and correlation analysis usually overpower individual fluctuations and often disclose more reliable hidden patterns and knowledge. Further, interconnected Big Data forms large heterogeneous information networks, with which information redundancy can be explored to compensate for missing data, to crosscheck conflicting cases, to validate trustworthy relationships, to disclose inherent clusters, and to uncover hidden relationships and models. Mining requires integrated, cleaned, trustworthy, and efficiently accessible data, declarative query and mining interfaces, scalable mining algorithms, and big-data computing environments [17];

5) Data interpretation arises from the need to give valuable information to most people as possible. The potential value of analyzing Big Data is only harnessed, if users understand the analysis. Therefore, data interpretation consists in transforming the existent data to a format that will be understandable for the ordinary user. In a nutshell, most of the time, just providing the results is not enough. Rather, supplementary information must be provided explaining how results were derived upon precisely what inputs. Such supplementary information is called the provenance of the resulting data. Improving the capture, storage and query provenance, in conjunction with adequate meta-data capture techniques, an infrastructure can be built, and provided to users, with the ability to interpret analytical results and to repeat the analysis with different assumptions/parameters or datasets; [1]

Although the analysis phase is crucial, it is important not to underestimate all other phases of the data analysis pipeline. For instance, Big Data needs to be managed in a context, which may be noisy, heterogeneous and not include an upfront model. Doing so raises the need to handle uncertainty and errors. Similarly, the questions to the data analysis pipeline will typically not all be laid out in advance. We may need to figure out good questions based on the data. Therefore, a good planning of the big picture, considering data acquisition, extraction,

integration, analysis and interpretation, must be done, in order to take most advantages of the provided data. [1]

Nowadays, there is a major bottleneck in the number of people capable to ask useful questions to the data and analyze it properly. It is possible to significantly increase the number of users capable to work with the data, by supporting many levels of engagement with it, not all requiring deep technical expertise. Existing computational techniques can be applied, at least in some aspects of the Big Data problem. For example, relational databases rely on the notion of logical data independence, therefore, users can think about what they want to compute, while the system determines the way it will be efficiently computed.

## 2.2 Business Intelligence

The ultimate benefit of Business Intelligence (BI) is to accelerate and improve decision making, with the intent of increasing operational efficiency, and eventually gaining competitive advantages over business rivals. It also helps to identify market trends and spot business problems early. BI data can be either dynamic or statical, since data can include historical information, as well as new data gathered from source systems as it is generated, enabling BI analysis to support both strategic and tactical decision-making processes. In the past, Business Intelligence tools were mostly used by data analysts and others IT professionals, but business executives are increasingly using BI software themselves, thanks to the recent development of user-friendly and ordinary user oriented BI platforms.

BI combines a broad set of data analysis applications, including ad hoc analysis and querying, enterprise reporting and online analytical processing (OLAP). BI technology also includes data visualization software for designing charts and other info-graphics, as well as tools for building BI dashboards and performance scorecards that display viewable data on business metrics and key performance indicators in an easy-to-grasp way. BI programs can also incorporate forms of advanced analytics, such as data mining, predictive analytics, text mining, statistical analysis and big data analytics. Figure 2.2 illustrates how the tools cooperate with each other in order to extract the potential business value of Big data.

BI environments are related with the management and processing of huge volumes of data, which must be stored efficiently, in order to smooth later analysis. Therefore, as Figure 2.3 suggests, data must be stored in specialized data structures, designed as data warehouses. [18]

A Data Warehouse usually encompasses a huge domain. However, that big domain is typically composed by smaller domains more or less independent between them. Each one of those small domains can be represented by a Data Mart. In other words, a Data Warehouse can be seen as a central repository for all data. Data Marts seek to meet the particular demands of a specific group of users. So, organization's Data Marts are subsets of the organization's Data Warehouse. [19]

In Data Warehousing, the simplest dimensional model is called the start schema, in this schema data are organized in facts and dimensions. A fact can be seen as an event, and a dimension contains information about the fact. A star schema is designed by surrounding each fact by the associated dimensions, the resulting schema resembles a star, as Figure 2.4 suggests, hence the schema name. Star schemes are optimized for querying large data sets and are used by Data Warehouses and Data Marts. The star schema boosts aggregation operations of the fact records, and those operations can be easily filtered or grouped by the surrounding dimensions. [20]

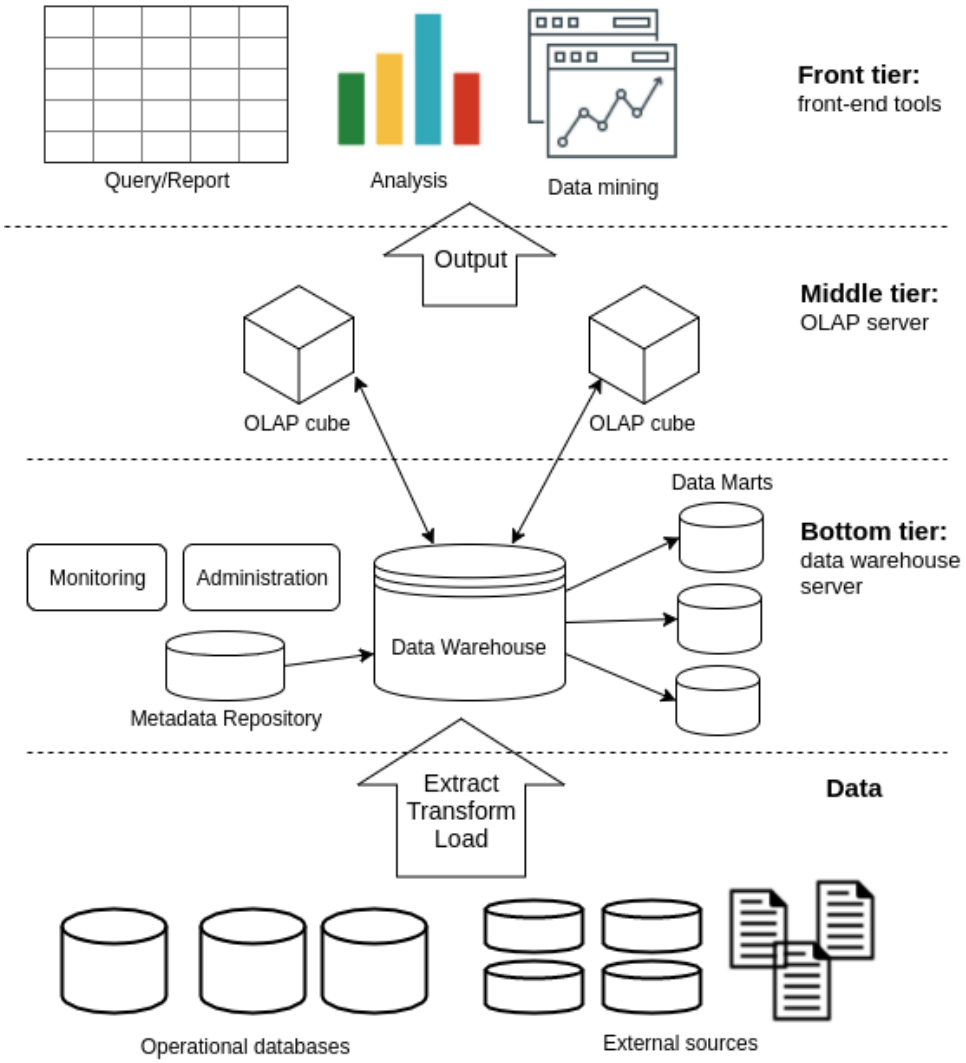



Figure 2.2: Business Intelligence Architecture.

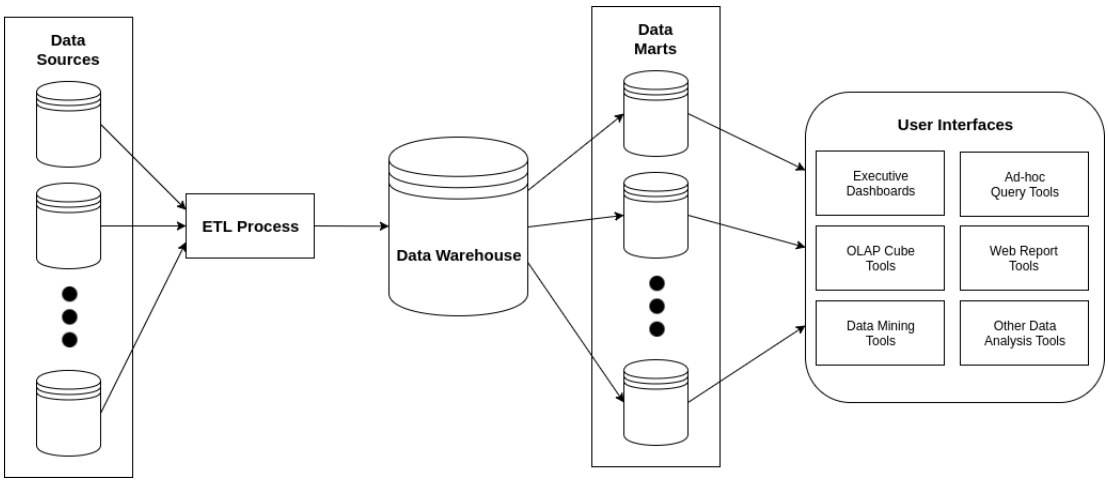



Figure 2.3: Business Intelligence Pipeline.

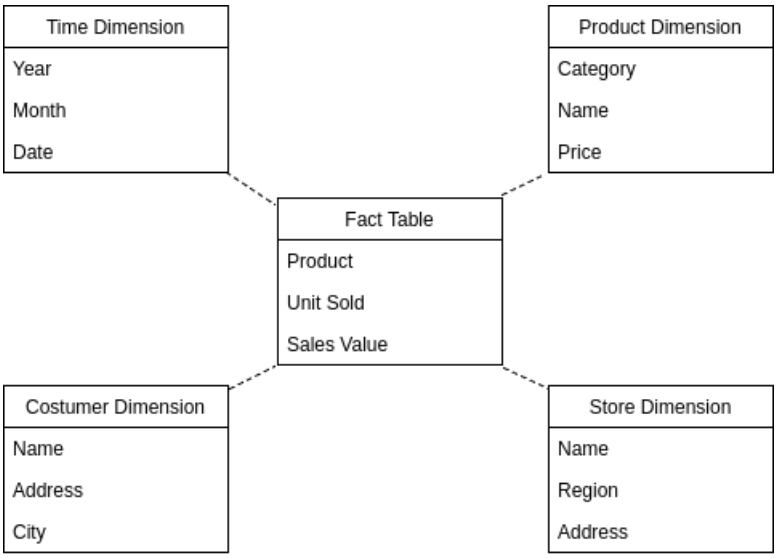



Figure 2.4: A Dimensional Model of a Data Warehouse.

To store data in an efficiently fashion within the Data Warehouse, and underlying Data Marts, the collected heterogeneous raw data must be properly parsed and handled. In this sense, Extract, Transform and Load (ETL) is a programming tool that combines three distinct functions. The extract function reads and collects data from distinct and heterogeneous specified data sources and extracts a subset composed by the useful data. The transform function manipulates the data to convert it to the desired form and state. And, the load function stores the resulting data to the target database.

OLAP is a technique that enables a user to easily and selectively extract and view data from a different point of view. OLAP data are stored in a multidimensional database, smaller than a data warehouse, since not all transactional data is needed for trend analysis. OLAP tools are designed to locate interceptions of the data dimensions and display them. [21]

In a business perspective, it enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user.

## 2.3 ETL tools

ETL tools are very useful for data integration and data warehousing. They act as basis for construction of data warehouses, since the input is given to the data warehouse through the ETL. There are a large number of ETL tools available in the market, following different designs and modeling techniques. Some are open source and others are proprietary tools. In this sense, is important to analyze and evaluate the most popular ETL tools available on the market. N. Mali et al. [11] and P. Metkewar et al. [12] published papers on ETL tools surveys, in order to understand and identify the most relevant ETL tools' pros and cons, and in what context they should be used. Briefly, both Talend Open Studio (TOS) and Pentaho Data Integration (PDI or Kettle) are by far the most relevant and complete open-source tools currently available on the market. Furthermore, Informatica PowerCenter and IBM Infosphere

Data Stage are the most popular commercial ETL tools.

Pentaho [1] is a commercial open-source Business Intelligence suite along with a data integration product named Pentaho Data Integration - PDI (Kettle). Kettle is the default ETL tool for the Pentaho ecosystem. It uses a meta-driven approach. Moreover, Pentaho provides a graphical editor, called Spoon, where users are able to build data integration procedures, also known as transformations. The transformations can be run by Kettle runtime in different ways: using the command line utility (Pan or Kitchen), a remote server (Carte) or directly from the IDE (Spoon). Procedures can be saved in a database repository or a file system, in XML format, and interpreted by a Java library which is required to run the ETL tasks. It also supports multi-format data and allows data movement between many different databases and files. These interactions are pictorially represented in Figure 2.5. [2]

Talend Open Studio is another open-source ETL tool, developed by Talend [2], with support of data integration. Rather than meta-data driven it uses a code-driven approach and has a user friendly and comprehensive GUI for user interaction, similar to Spoon. The code generation property allows generating executable code of Java and Perl that can be run later on a server.

Informatica Power Center [3] (IPC) is a commercial data integration suite, and share leader in data integration tools. [22] The main focus of IPC is on data integration with numerous capabilities, namely: enterprise size architecture, data cleansing, data profiling, web servicing and interoperability with current and legacy systems.

IS Datastage [4] is a IBM product. The capabilities of the tool include data consolidation, synchronization, and distribution across disparate databases, automatic data profiling and analysis in terms of content and structure, data quality enhancement, transformation and delivery to and from complex sources, i.e. capability to get data from any sources format and deliver it to any targets, within or outside the enterprise, at the right time.

### 2.3.1 Open-Source Tools

Both Talend and Pentaho have the largest communities and strongest community support. In my own perspective, I felt that Pentaho's community is more active and helpful than Talend's community. Nevertheless, Talend and Pentaho represents the most deployed open-source ETL tools. Talend is more focused on data integration, data quality and data management solutions, while Pentaho is focused on Business Intelligence. Pentaho and Talend solutions are very reliable, mature and fast growing. Real world enterprise implementations are becoming common in both cases.

Talend's components and features are numerous, mixing both general purpose tools and very specific components. Talend provides vendor specific sets of RDBMS, NoSQL, Big Data components among generic ones, this approach enables the support to both vendor specific features and generic database features. Pentaho's features and components are a little less comprehensive than Talend ones, however this doesn't restrict the complexity of the ETL procedures that can be implemented. In fact, the most complex and specific Talend's components can be built by a set of less complex and more general Pentaho's components.

[1] https://www.hitachivantara.com/go/pentaho.html?source=pentaho-redirect
[2] https://www.talend.com/
[3] www.informatica.com
[4] https://www.ibm.com/us-en/marketplace/datastage

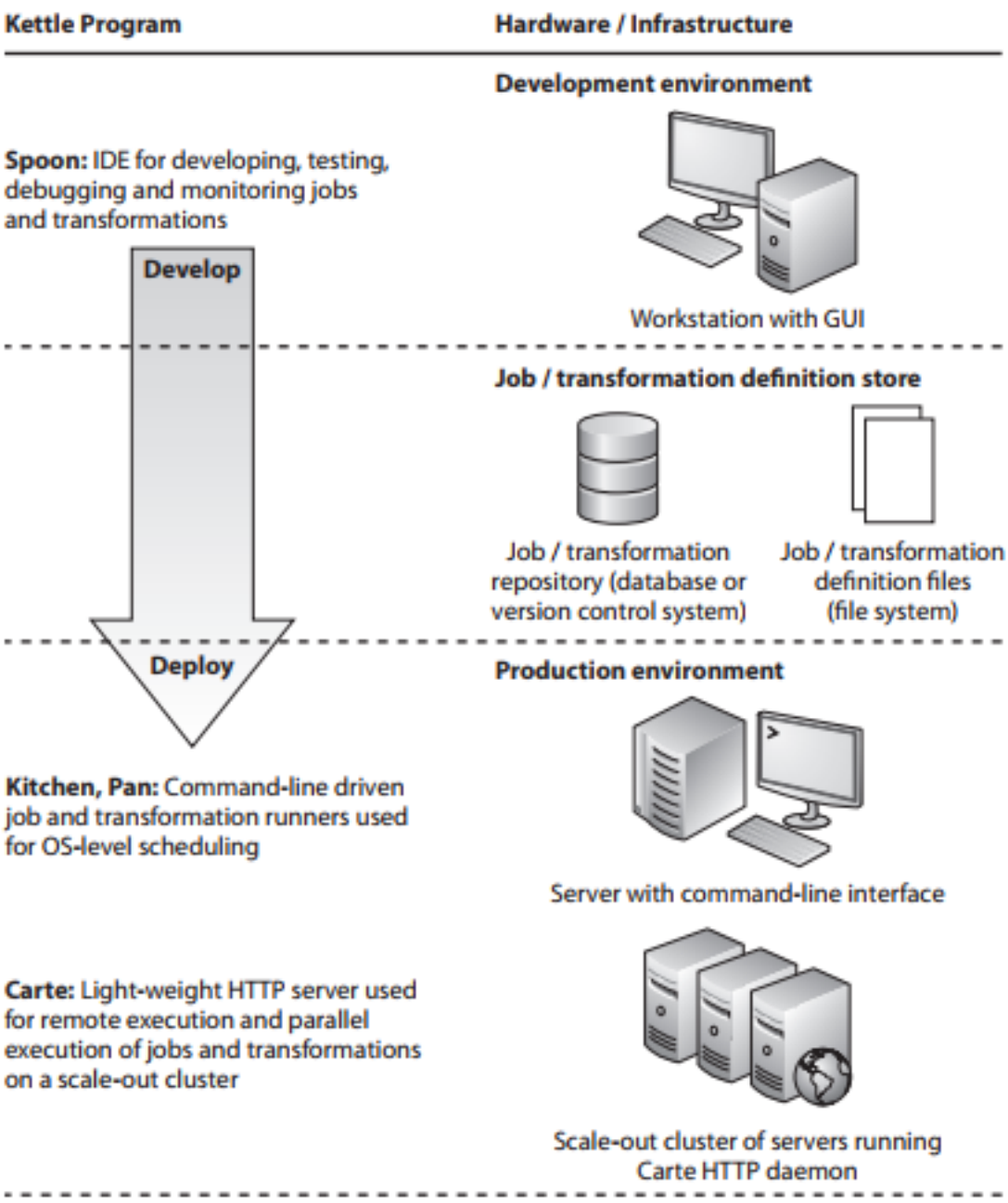



Figure 2.5: Overview of Kettle programs. Source by [2]

Talend is very easy to deploy as a standalone Java application, and is the default ETL tool for SpagoBI and JasperServer. Kettle is the default tool for Pentaho Business Intelligence and requires the Kettle core libraries when deployed outside the Pentaho platform. However, Kettle makes it easy to deploy procedures in clustered environments and save them in a database table. In Talend these and further features are available in the subscription version.

Hermanny Filho [23] performed a comparative analysis between Pentaho Data Integration and Talend Open Studio. Both tools were compared within three distinct categories: the way transformations and jobs are developed, the available features by the tools and the transformations' performance in both tools. The author concluded that Kettle presented better results on the flexibility to develop transformations or jobs and on the data migration speed. On the other side, Talend Open Studio made less memory and CPU usage on the data movement tasks. In more detail, in the best scenario, Kettle showed 60% quicker than Talend Open Studio, on the data movement tasks. While, in the worst scenario, used 80% more CPU resources.

Majchrzak et al.[24] published a paper on ETL applications and in particular on open source ETL tools. That paper analyzes open-source ETL tools especially with regard to their performance. Before the selected ETL tools' performance evaluation, the authors evaluate the potential tools, by a criterion. Since Pentaho Data Integration and Talend Open Studio were the ones that fulfill the criterion, as Table 2.1 suggests, they were the chosen tools for the performance analysis. A selection of common indicators was derived from the ISO 9126 norm, to evaluate the performance, namely: the execution time, measured in seconds, the CPU load, measured in percentage of utilization, the memory load, measured in percentage of utilization and the number of SELECT SQL statements.

Table 2.1: Criteria fulfillment by the candidates

| | Integration | Interfaces | Graphical editor | Functionality | Support | Documentation | Up-to-dateness |
|---|---|---|---|---|---|---|---|
| Apatar OS Data Integration | | X | X | X | X | X | X |
| CloverETL Engine | X | X | | X | X | X | X |
| Enhydra Octopus | X | X | | X | | | |
| Jitterbit Integration Environment | | X | X | X | X | X | X |
| KETL | X | X | | X | | X | X |
| Pentaho Data Integration | X | X | X | X | X | X | X |
| Scriptella | X | X | | X | | X | |
| Talend Open Studio | X | X | X | X | X | X | X |

After the performance evaluation, the authors found that Talend Open Studio(TOS) is twice as fast, regardless of the amount of data. And that the speed advantage of TOS varies around a factor of six. There is also a slightly worse scaling behavior in the CPU usage of PDI compared to TOS.

On the other side, the memory usage of PDI is advantageous. It is about 40-50% less than the average memory usage of TOS, regardless of the amount of data processed. Moreover, TOS does not support the parallel organization of the ETL process.

To sum up, the claim of completeness and consistency is fulfilled by TOS and PDI. While for TOS low CPU utilization is given in most cases, PDI has consistent advantages in memory usage. Even with varying memory size both tools show similar behavior.

Nonetheless is of major importance to realize that the transformation's implementation plays a huge role on the execution performance of the actual transformation. Overall, it may be said both tools are complementary. Each one with a focus, but allow the same tasks of transformation and data integration.

### 2.3.2 Open-Source tools vs Commercial tools

Marc Russel made a comparative analysis between the major open-source and commercial ETL tools (PDI vs TOS vs IPC vs DataStage). [25] The analysis focused on performance and usability. The author highlighted the difficulty of providing a reliable analysis. But the global performance of each ETL tool was scored, as a sum of twelve distinct performance tests. Thus, the following scores were obtained:

- First: Informatica Powercenter (353 points)
- Second: Talend Open Studio (333 points)
- Third: IBM Datastage (239 points)
- Fifth: Pentaho Data Integration (148 points)

Nevertheless, PDI is the easier tool to parallelize ETL workloads.

R. Katragadda et al. performed an empirical study of Talend Open Studio versus Informatica Power Center [26]. The authors found out that IPC has multiple capabilities, but limited functionalities in integration with other environments. Whereas Talend has advanced integration capabilities with other environments. Furthermore, Talend is quite suitable for small implementations, but for large implementation IPC is way ahead. Since Talend is free, its support, documentation and large-scale implementations make it less suitable for commercial application.

H. Rose et al. [27] made a comparative study between open-source and commercial ETL tools. The authors evaluated ETL tools in terms of architecture, functionality, usability, reusability, connectivity and interoperability. From all the analysis conducted it is still hard to generalize which tool is the best. However, commercial tools proved to be generally better. Infomatica proves to be slightly better in quite many features. Nevertheless, the authors emphasized that open-source tools offer reduced costs, having the potential to give support to the initiatives of small enterprises. Furthermore, the gap between open-source and commercial ETL tools has been decreasing over the time.

## 2.4 Business Analytics Tools

Data visualization is the presentation of data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns. With interactive visualization, you can take the concept a step further by using technology to drill down into charts and graphs for more detail, interactively changing what data you see and how it's processed.

Charts and graphs represent in a more intuitive fashion large amounts of complex data, rather than spreadsheets or reports. With the aid of data visualization, decision-makers can identify areas of attention or improvement in the business, clarify which factors influence customer behavior, analyze and predict the best place for a specific product and predict sales volumes.

In this sense, is important to analyze and evaluate the most popular business analytics tools available on the market.

### 2.4.1 Power BI

Power BI [5] is a suite of business analytics tools to analyze data and share insights, provided by Microsoft. It offers interactive visualizations with self-service business intelligence capabilities, where end users can create reports and dashboards by themselves, without having to depend on information technology staff or database administrators. Power BI ecosystem encompasses a set of key components:

- **Power BI Desktop:** a windows desktop application.
- **Power BI (Service):** a SaaS (Software as a service) online service.
- **Power BI Mobile Apps:** an application for Android and iOS devices, as well as for Windows phones and tablets.
- **Power BI Gateways:** gateways used to sync external data in and out of Power BI.
- **Power BI Embedded:** Power BI REST API, that can be used to build dashboards and reports into the custom applications that serves Power BI users, as well as non-Power BI users.
- **Power BI Report Server:** provides a localized way for storing and managing Power BI reports.
- **Power BI Visuals Marketplace:** a marketplace of custom visuals and R-powered visuals.

Figure 2.6 portrays how Power BI tools are orchestrated, to cooperate between them, in order to build and deliver business insights. Within the Power BI environment, there are hundreds of connections to popular business applications, for instance Excel, Google Analytics, Salesforce or MySQL, complete with prebuilt dashboards. Also, data and reports are accessible anywhere with Power BI Mobile apps, that can synchronize automatically on data changes. Moreover, Power BI allow data analysts to build reports and analytics, over Power BI Desktop. It combines data from disparate databases, files, and web services with visual tools that helps to understand and fix data quality and formatting issues automatically. Furthermore, reports can be securely published and automatically refreshed, within the organization. Power BI can unify all organization's data, with Power BI gateways. Finally, Power BI reports and dashboards can be embedded in external reporting portals or applications, with Power BI Embedded.

### 2.4.2 Metabase

Metabase [6] is a free, open source analytical tool that allows the user to ask questions about the provided data, without SQL knowledge, through a very simple web interface. The asked question is converted to SQL code, through a powerful translator. Results are retrieved in table format, where users can re-filter the result or translate the table into several different graphs. Latter, dashboards can be built by dragging and resizing multiple resulting graphs into a mesh (Figure 2.7).

[5] https://powerbi.microsoft.com/en-us/
[6] https://www.metabase.com/

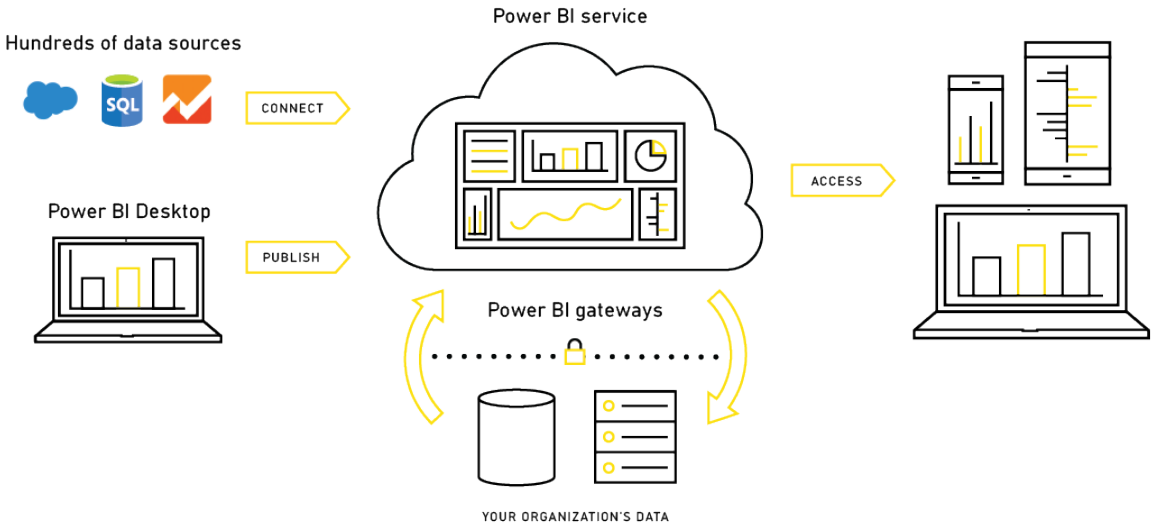



Figure 2.6: Power BI ecosystem.

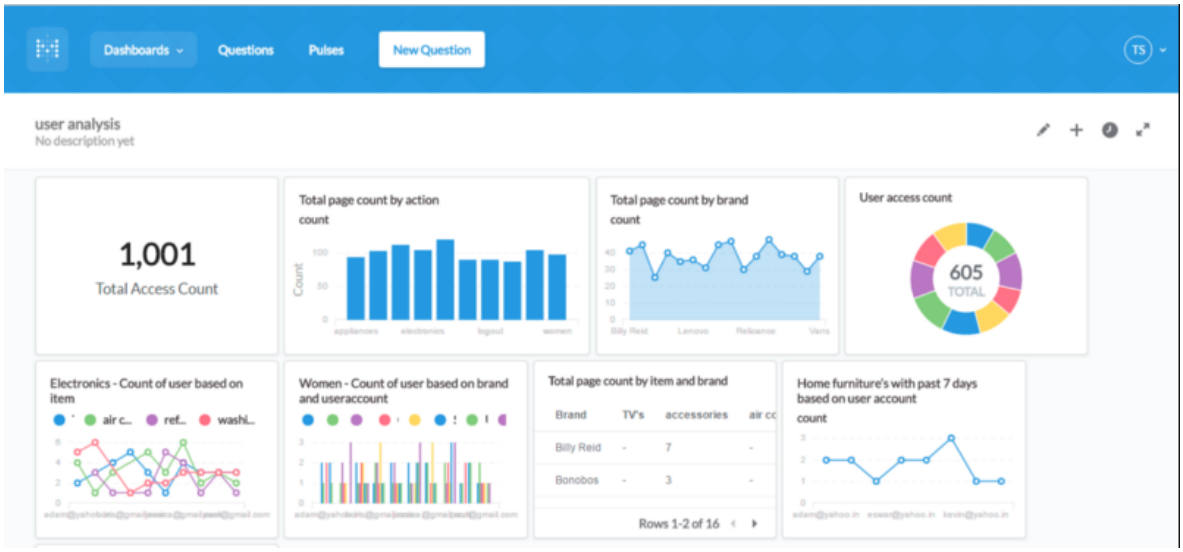



Figure 2.7: Metabase dashboard.

Metabase offers advanced capabilities, such as sending result notifications via email or Slack, embedding the resulting dashboards on external web pages and geographic maps construction. Moreover, Metabase allows fully customizable roles management and has a modern feed wall, where users can be informed of colleagues' activities. Furthermore, it also provides a REST API, where developers can interact with the engine.

### 2.4.3 Saiku

Saiku [7], originally called the Pentaho Analysis Tool, started life as a basic Google Web Toolkit(GWT) based wrapper around the OLAP4J library. Saiku offers a user friendly, web-based analytics solution that allows to easily analyze corporate data and share reports. The solution connects to a range of OLAP Servers including Mondrian, Microsoft Analysis Services, SAP BW, allowing to explore data in real time. By harnessing the power of OLAP, Saiku allows users to choose the measures and dimensions they need to analyze, split data and drill into the detail to uncover relationships, opportunities and issues. Furthermore, allows non-destructive editing of query results, providing the ability to adjust the figures and perform "what-if" analysis over their data (Figure 2.8). The query results can be shared or exported into Excel or PDF.

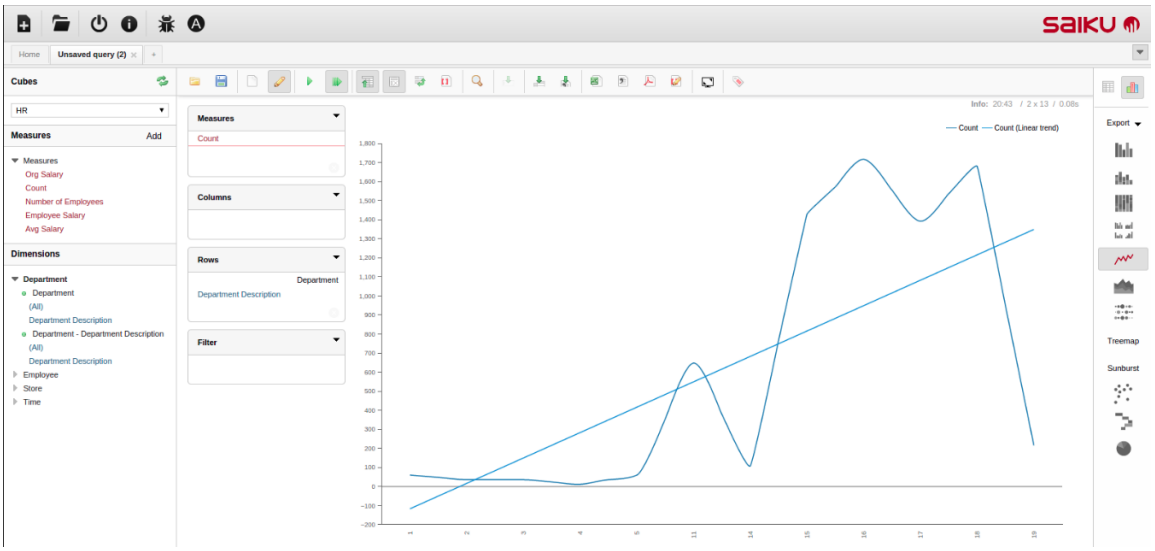



Figure 2.8: Saiku dashboard.

Saiku can be embedded in external web applications. Because of its restful nature, the server can be completely decoupled and used as a standalone service. Moreover, Saiku Embed Framework is a framework that facilitates Saiku's integration within external applications.

## 2.5 Business Intelligence Suites

Business Intelligence encompasses a wide variety of tools, applications and methodologies that enable organizations to collect data from internal systems and external sources, prepare it for analysis, develop and run queries against that data, and create reports, dashboards and

[7] https://meteorite.bi/products/saiku

data visualizations. The anlytical results are then available to corporate decision-makers, as well as operational workers.

Since business intelligence requires a set of tools or technologies with different goals or purposes, those tools are traditionally bundled. The BI bundle is called Business Intelligence Suite. Thus, is important to analyze and evaluate the most relevant BI Suites available on the market.

### 2.5.1 Pentaho

Pentaho is a complete business intelligence suite. Pentaho BI Project encompasses the following major applications: Reporting, Analysis, Dashboards, Data Mining and Business Intelligence Platform. Pentaho BI Suite consists in Data Integration (DI) and Business Analytics (BA) components. All core engines are open. As Figure 2.9 suggests, Pentaho's stack includes three layers:

- **Presentation layer:** responsible for news, analysis, dashboards and method management. The data can be viewed either from a browser, portal, office, and e-mail or web services.
- **Business Intelligence Platform:** responsible for security, administration, business logic and repository management.
- **Data & Application Integration:** responsible for ETL and its integration.

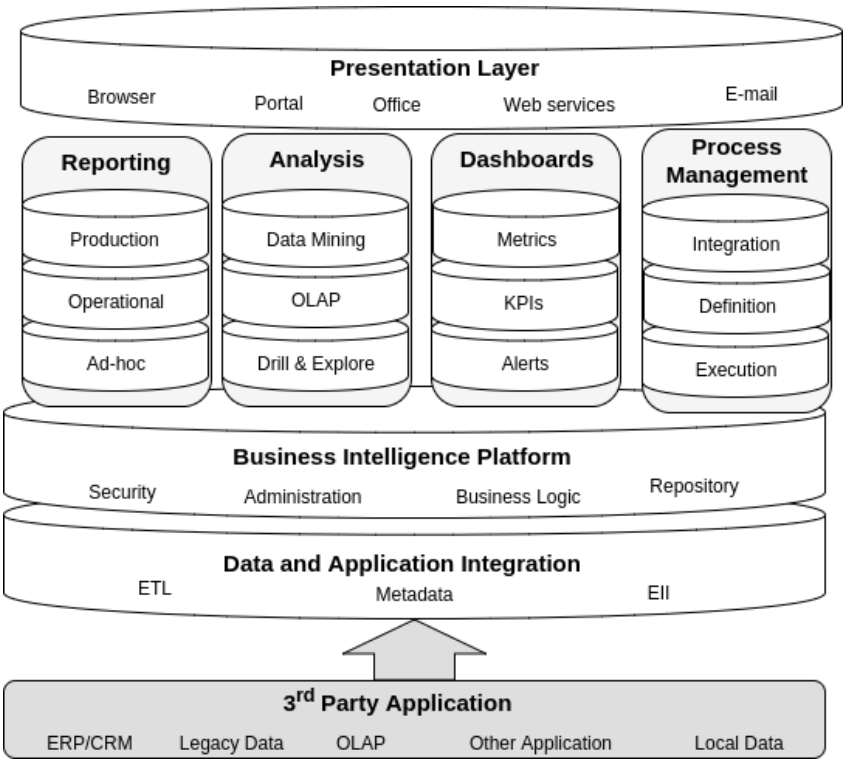



Figure 2.9: Pentaho Business Intelligence Suite.

From a architectural point of view (Figure 2.10), Pentaho also contains three layers:

- **Data Layer:** allows the connection to any data source.
- **Server Layer:** encompasses BA Server, Pentaho User Console and Pentaho Data Integration Server. User console is for the user role mapping, security and for configuring which report should be shown to which user. Pentaho Data Integration server handles jobs and transformations' execution. Reports and dashboards can be deployed via BA server, making them available to the end-users.
- **Client Layer:** offers a set of front-end tools for corporate data manipulation and analysis, and for reports and dashboards construction.

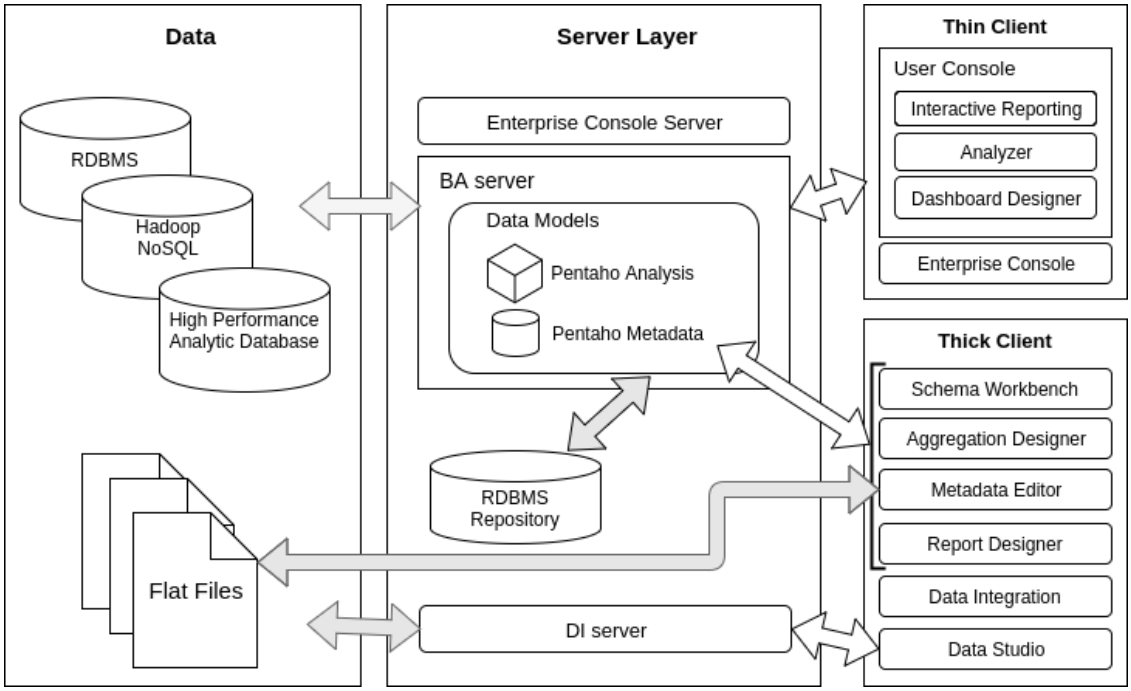



Figure 2.10: Pentaho Architecture.

### 2.5.2 SpagoBI

SpagoBI [8] contains a variety of functionalities such as the creation and export of reports, creating charts, ETL, OLAP, data mining, KPIs, application server, login service, dashboard, task schedule and Web server. All modules are connected with the core, which ensures the balance of the platform. Figure 2.11 illustrates SpagoBI Server architecture. SpagoBI can be represented by five components:

- **SpagoBI Server:** the core of the suite including the analytical tools and features.
- **SpagoBI Meta and Studio:** the integrated development environments to organize and create analysis.

[8] https://www.spagobi.org/

- **SpagoBI SDK:** the integration layer allowing to use SpagoBI with external tools.

- **SpagoBI Applications:** a collection of vertical analytical models that are developed using SpagoBI.

Beyond the Business Intelligence server, SpagoBI offers a software development kit (SDK) used for the integration of the services provided by the Server. In particular, it is used by SpagoBI Studio in order to allow users to download or upload their analytical documents from or onto the Server.

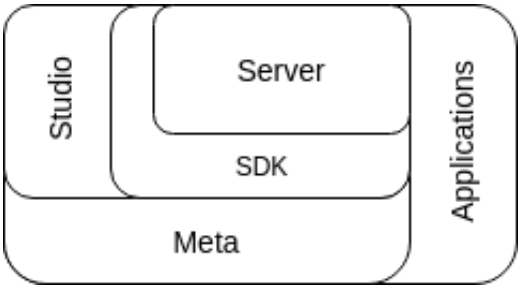



Figure 2.11: SpagoBI Architecture.

### 2.5.3 Jaspersoft

Jaspersoft [9] BI Suite consists of five different components (Figure 2.12):

- **JasperReports Library:** the reporting engine, is able to use different types of data source to produce reports that can be viewed, printed or exported in several documents formats, including HTML, PDF, Excel and Word.

- **Jaspersoft Studio:** is the editing software for JasperReports. It allows to design and run report templates, build report queries and write complex expressions.

- **JasperReports Server:** the reports' server, responsible for storage and report management.

- **Jaspersoft ETL:** the ETL software with a GUI. It allows the data extraction from your transactional system to create a consolidated data warehouse or data mart for reporting and analysis.

- **Jaspersoft OLAP:** the relational OLAP server. It enables data analytics to model, manipulate and visualize any flavor of data using OLAP or in-memory analysis in order to identify issues, spot trends and make better decisions quickly.

[9] https://www.jaspersoft.com/

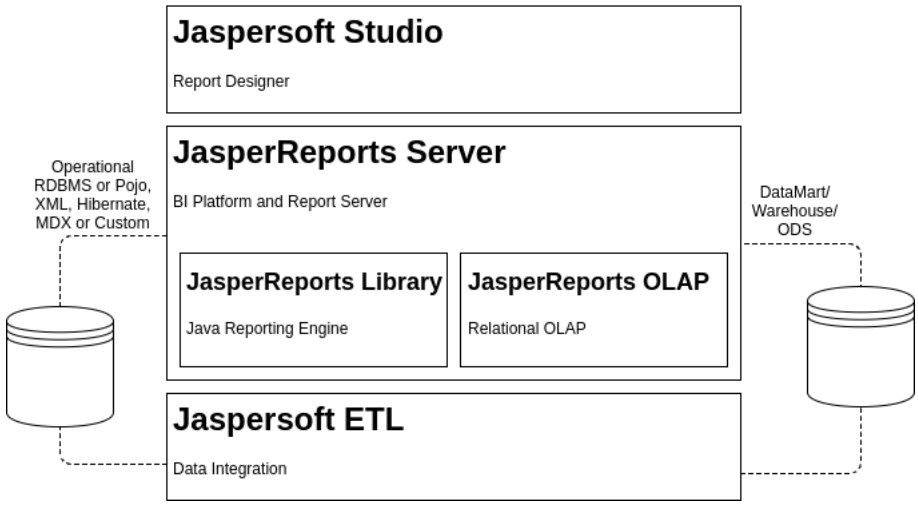



Figure 2.12: Jaspersoft Architecture.

### 2.5.4 BI Suites comparison

J. Lapa et al. [28] performed a comparative analysis between several open-source business intelligence platforms: Actuate, JasperSoft, OpenI, Palo, Pentaho, SpagoBI and Vanilla. The authors evaluated the BI platforms according to the criteria used in Gartner. These criteria are grouped into three broad and fundamental categories: Integration, Information delivery and Analysis. They didn't evaluate the usability, nor the platforms' performance. Instead they focused on revealing each platform's capabilities. After the evaluation, they concluded that Pentaho, SpagoBI and Jaspersoft are the most complete open-source BI platforms currently available.

V. Parra et al. [29] realized a comparative study on the data processing performance of Pentaho and Jaspersoft, in Big Data scenarios. The experimental analysis has focused on their ETL and reporting processes, by measuring their performance. Six different Excel databases, with different sizes, have been used to perform the analysis. In the ETL phase, Jaspersoft BI revealed an increment on the data processing CPU time over Pentaho BI, represented in an average of 42.28% over the six databases. Meanwhile, Pentaho BI exhibited an increment on data processing CPU time over Jaspersoft evidenced by the reporting analysis outcomes, with an average 43.12% over the databases. In summary, Pentaho had a superior performance for the ETL process, and Jaspersoft an improved performance for the reporting process.

A. Marinheiro et al. [30] elaborated a study that evaluated and compared the five main open-source Business Intelligence suites: JasperSoft, Palo, Pentaho, SpagoBI and Vanilla. The authors identified Pentaho and SpagoBI has the most powerful solutions. Nevertheless, Pentaho has a commercial version and doesn't allow collaborative work. However, it is the most used open-source BI suite. The major disadvantages on Jaspersoft is that, in its open-source version, doesn't support ad-hoc queries, data-mining and score-boarding. Nonetheless, those capabilities are fulfilled in the commercial version. Finally, they concluded that currently open-source BI suites offer a high level of confidentiality, and should be considered as valid alternatives to commercial BI suites.

## 2.6 Summary

In this chapter, it was highlighted the importance of supporting many levels of engagement with the data, so that a higher number of users are able to take advantage from it. There are multiple types of tools which help users to take full advantage of data. ETL tools simplify the data integration process for the construction of a data warehouse. BA tools provide means for representing data intuitively. Tipically, these tools are bundled as a BI suite. Therefore, a comparative study was performed to identify the best current BI suite. Briefly, Pentaho proved to be the best open source BI suite.

# Chapter 3

# Proposal Definition

The main goal of this thesis was to build a web-based platform that allows the building and management of ETL pipelines, by non-IT users, in a multi-institution environment. In this section, it will be presented the requirements that were considered to implement a proper and reliable solution.

## 3.1 Functional Requirements

Each institution manages and mantains ETL tasks and provides the resources for the execution of the associated tasks. Thus, each institution owns its private data sources, servers for ETL task execution and a task scheduler that allows periodic execution. In order to provide access and management control of ETL tasks and institutions, there are four distinct types of users (Figure 3.1):

- **Administrator:** Entity that moderates the platform. This actor has permissions to create and delete institutions.
- **Resource Manager:** Entity that manages private data sources and execution servers. This actor has permissions to create and delete private data sources and execution servers, within specific institutions.
- **Task Manager:** Entity that builds and executes ETL tasks. This actor can create and configure ETL tasks, within specific institutions.
- **Data Analyst:** This actor has permissions to inspect task execution history, namely the resulting data, execution logs and performance metrics.

The main use cases are represented in Figure 3.1, where are represented the user interactions with the web application and institutions. Table 3.1 identifies the main actor for each use case and describes it as a user story.

To enable the development of ETL tasks, the platform must provides a set of ETL components. The platform should provides 12 different ETL components divided in 7 distinct categories (Table 3.2).

Finally, users that belongs to the institution must have access to that institution in the system. Thus, upon authentication the system must be able to authenticate users via institution Lightweight Directory Access Protocol (LDAP) server or Active Directory (AD) server.

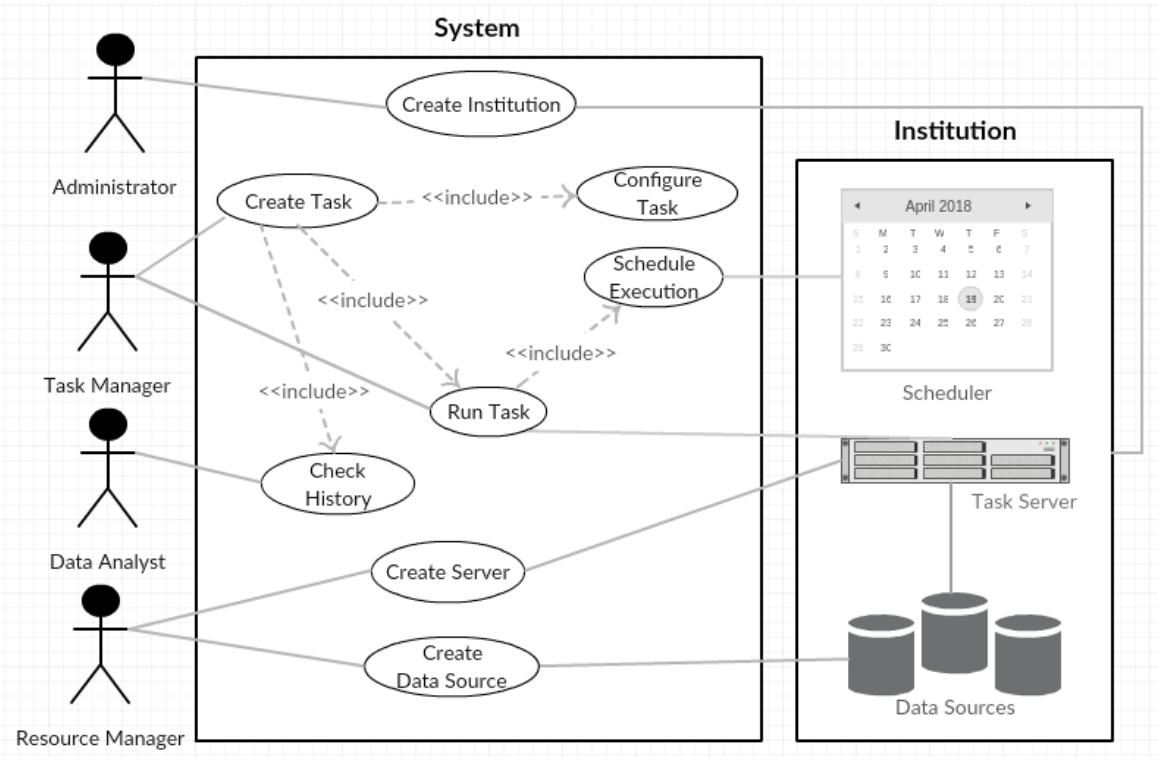



Figure 3.1: System use cases.

Table 3.1: List of user stories

| Primary Actor | Use Case | User Story |
|---|---|---|
| **Administrator** | Create Institution | As an administrator,<br>I want to create a institution,<br>So that other users can build, execute and manage ETL tasks. |
| **Resource Manager** | Create Server | As a resource manager,<br>I want to create a task server,<br>So that task managers can execute ETL tasks. |
| | Create Data Source | As a resource manager,<br>I want to create a data source,<br>So that ETL tasks can access institution's private data. |
| **Task Manager** | Create Task | As a task manager,<br>I want to create a task,<br>So that it can be correctly configured. |
| | Run Task | As a task manager,<br>I want to run a task,<br>So that the data is processed in some way. |
| **Data Analyst** | Check History | As a data analyst,<br>I want to check a task's history,<br>So that I can chronologically verify executions' output and performance metrics. |

Table 3.2: List of supported ETL components

| Category | Component | Description |
|---|---|---|
| **Input** | Generate Rows | Generate a number of empty or equal rows. |
| | Table Input | Read information from a database table. |
| **Ouput** | Table Output | Write information to a database table. |
| **Transform** | Add a checksum | Add a checksum column for each input row. |
| | Select values | Select or remove fields in a row.<br>Optionally, set the field meta-data: type, length and precision. |
| | Add sequence | Get the next value from an sequence. |
| | Sort Rows | Sort rows based upon field values (ascending or descending). |
| **Flow** | Filter Rows | Filter rows using simple equations. |
| | Dummy (do nothing) | This step represents a breakpoint in the pipeline.<br>It's useful for testing purposes or to split streams. |
| **Join** | Merge Join | Joins two streams on a given key and outputs a joined set.<br>The input streams must be sorted on the join key. |
| **Statistics** | Group By | Builds aggregates in a group by fashion. |
| **Scripting** | Execute SQL script | Execute an SQL script, optionally parameterized using input rows. |

## 3.2 Non Functional Requirements

### Information Security

Since ETL tasks parse and handle sensitive data that belongs to a particular institution, the system must be designed and implemented taking in account these security issues, namely user authentication, access control, data protection and isolation.

### System Reliability

Considering the periodic execution of ETL tasks, it is important to ensure that each execution is correctly initialized, started, motorized and concluded. When some fatal error occurs during an ETL task execution, the system must be able to handle the error and successfully conclude the execution.

### Solution Scalability

Since a complete ETL tool typically encompasses a wide variety of components, it is crucial to build an agile approach to the development and integration of new ETL components.

## 3.3 System Modeling

In the following, the section system will be conceptualized according to the requirements identified in previous sections.

### 3.3.1 Functional Model

The functional model was derived from the identification of all key components from a top-down analysis of the system requirements. Figure 3.2 illustrates step by step all actions and interactions between the main actors in order to build and execute ETL tasks and to analyze the output results. Firstly, the administrator must create an institution. A resource

manager with access to the institution can create and configure all desired private data sources and task execution servers. Thereafter, a task manager can build and configure the ETL task and then schedule it for periodic or non-periodic execution. At the configured times the task will be sent to the execution server and executed. At the end of the execution, a notification will be sent to the data analysts, so they can check the execution results.

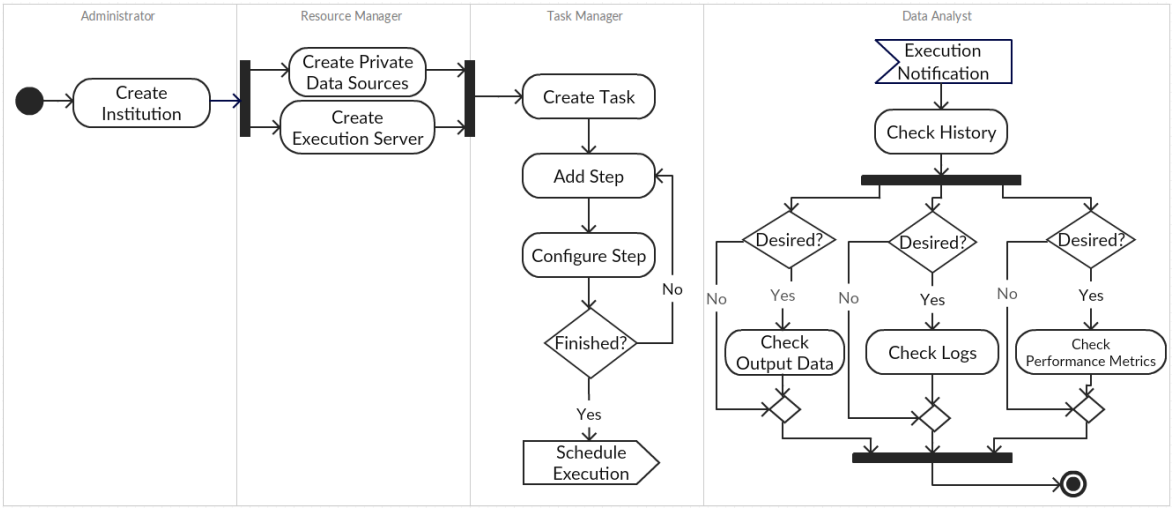



Figure 3.2: System workfow to build, manage and execute ETL tasks.

### 3.3.2 Data Model

The system data model relies on a SQL-structured database. The following subsection will detail the data model. Due to its complexity, the entity-relationship (ER) model will be explained in four steps. First, will be presented the ETL task structure. Second, will be introduced the institution structure and the underlying resources. After, will be explained how the results and metrics are organized. Finally, will be presented the role-based access control (RBAC) structure.

#### ETL process

An ETL process is represented by a pipeline of steps, that execute a certain transformation over the data stream, and hops, that connects the steps between each others (Figure 3.3).

As Figure 3.4 suggests, an ETL process or task is formed by multiple Steps and Hops. Each Hop starts from a source Step and ends in a target Step and their role is to guide the data over the flow. Each Step defines a transformation action and that action can be configured through the web interface. On one hand, in order to visually represent each Step, they must hold a Cell that is formed by a certain coordinate and dimension. On the other hand, a Step follows a given structure or schema that is defined by the Component. A Component is composed by a set of properties, called ComponentProperty. Each ComponentProperty has a label, a type, and a source, to dynamically build the step settings page and the Kettle's method name to invokeStepPorperty contains the actual value of the Step's ComponentProperty. In fact, in order to materialize a given Step, a StepProperty must be defined by each Component-Property of the correspondent Step's Component. Nevertheless, if the ComponentProperty

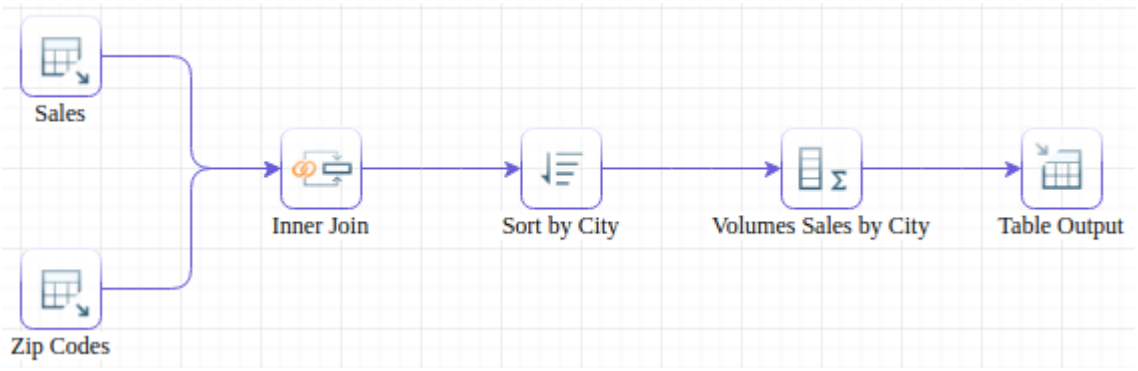



Figure 3.3: Example of an ETL process.

is a table, the property will be considered as a aggregate of sub-properties. In this case the ComponentProperty will have a set of ComponentMetadatas. Each ComponentMetadata will represent a column of the table and the same attributes will apply to ComponentMetadata.

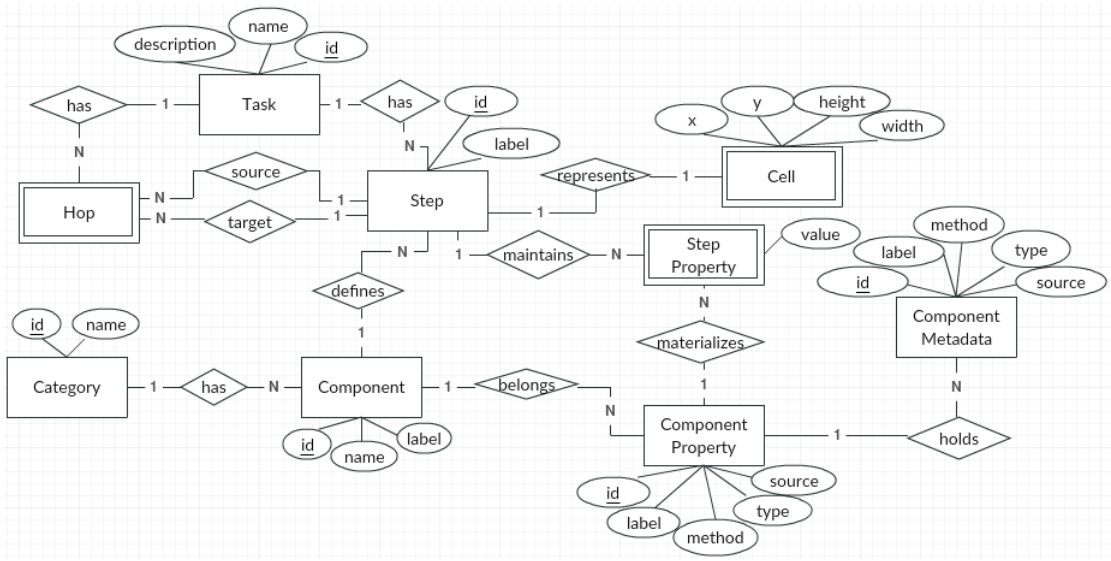



Figure 3.4: ETL process ER model.

### Institution

An institution is characterized by a name, a group of ETL tasks, a group of private data sources, a group of execution servers and a group of users (Figure 3.5). A data source consists in a relational database. Table 3.3 describes all data source attributes. Table 3.4 explains all atributtes of the remote execution server. To schedule task executions, Schedule entity relates a task, with a given server and user that scheduled execution. It also has a start date and a period that defines the interval between executions (Daily, Weekly, Monthly, Yearly).

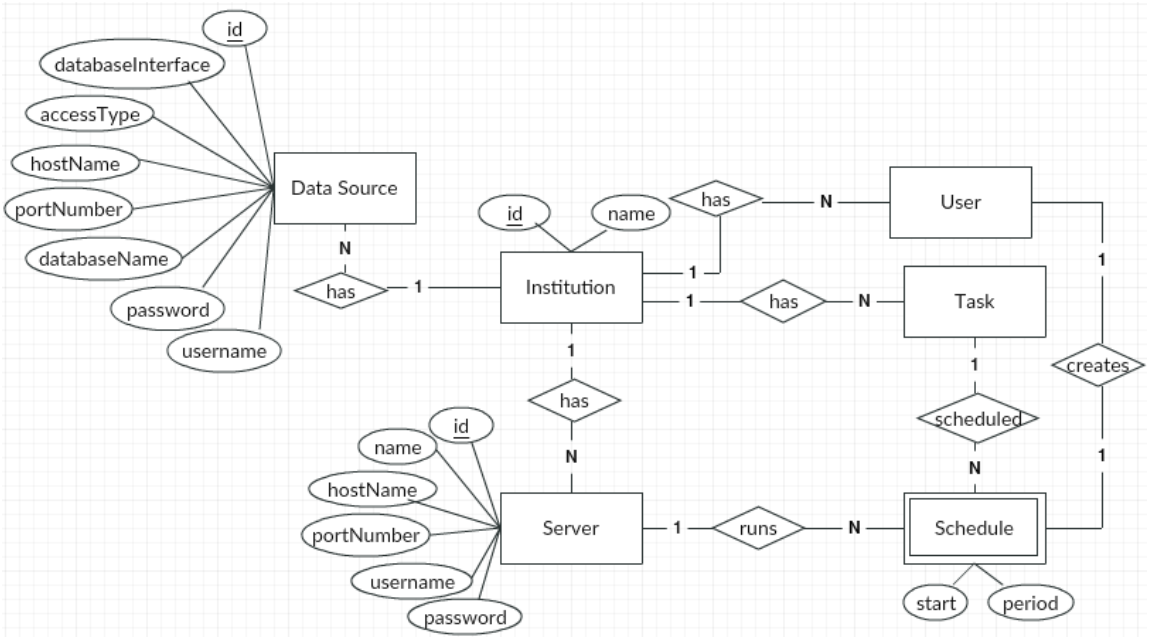



Figure 3.5: Institution ER model.

Table 3.3: Data source fields

| Field | Feature | Description |
|---|---|---|
| connectionName | Connection Name | Uniquely identifies a connection across transformations. |
| databaseInterface | Connection Type | Database technology (for example, MySQL, Oracle, and so on). |
| accessType | Method of access | Database access type (Native (JDBC), ODBC, or OCI).<br>Available access types depend on the database technology. |
| hostName | Server host name | Defines the host name of the server on which the database resides.<br>The host can also be defined by IP-address. |
| portNumber | Port number | Sets the TCP/IP port number. |
| databaseName | Database name | Identifies the database name to be connected.<br>In case of ODBC, specify the DSN. |
| username | Username | Specifies the user name to connect to the database. |
| password | Password | Specifies the password to connect to the database. |

Table 3.4: Execution server fields

| Field | Feature | Description |
|---|---|---|
| name | Server name | The alias of the execution server. |
| hostName | Hostname or IP address | The address of the machine where the server resides. |
| portNumber | Port number | Defines the port to use for communicating with the remote server. |
| username | Username | Username credential for accessing the remote server. |
| passwrod | Password | Password credential for accessing the remote server. |

### Output and Performance

When execution time comes, the task will be sent to the remote execution server and will be executed. Execution entity represents the task execution. An execution is formed by a set of step metrics and the output data, that holds the resulting data. Table 3.5 describes the step metrics in detail. The output data consists in a set of data rows. Each data row has a set of key-value pairs (Figure 3.6).

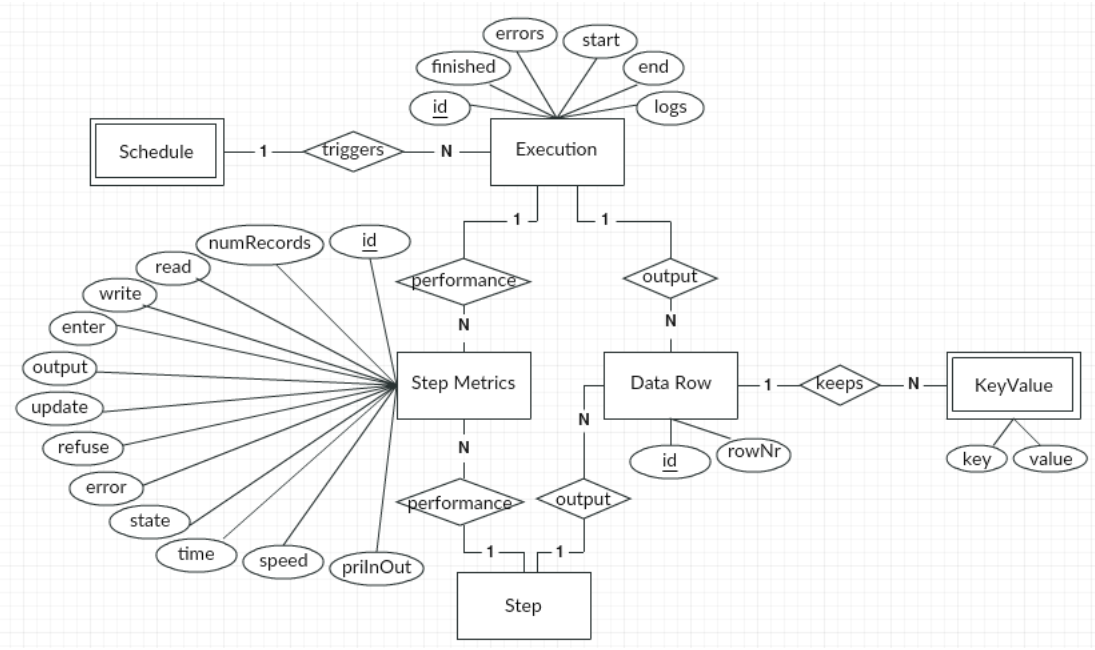



Figure 3.6: Execution ER model.

Table 3.5: Step Metrics fields

| Field | Feature | Description |
|---|---|---|
| read | Read | Number of lines read from input-streams. |
| write | Written | Number of lines written to output-streams. |
| input | Input | Number of lines read from file or database. |
| output | Output | Number of lines written to file or database. |
| update | Updated | Number of lines updated in the database. |
| error | Errors | Number of errors that occurred. |
| state | Active | The status of the step: running, finished or stopped. |
| time | Time | The number of seconds that the step has been running. |
| speed | Speed | The speed in rows per second at which the step processes rows. |
| priInOut | Input/Output | Sleep time (get/put) is the time that a step had to go to sleep (in nano seconds) because the input buffer was empty (get) or the output buffer was full (put). |

**RBAC**

Figure 3.7 describes the RBAC system structure. A user has several roles. Each role maintains a set of permissions. Each permission consists in an association of an operation and a resource. The Authentication entity is a facade to a given user group, that can be LDAP or Active Directory (AD). When a certain user logs in the platform the underlying user group will be determined, by trying authentication on each configured user group. If authentication succeeds, the user will be instantiated in the database. Depending on the group to which he belongs, the user will acquire the correspondent roles and institution access.

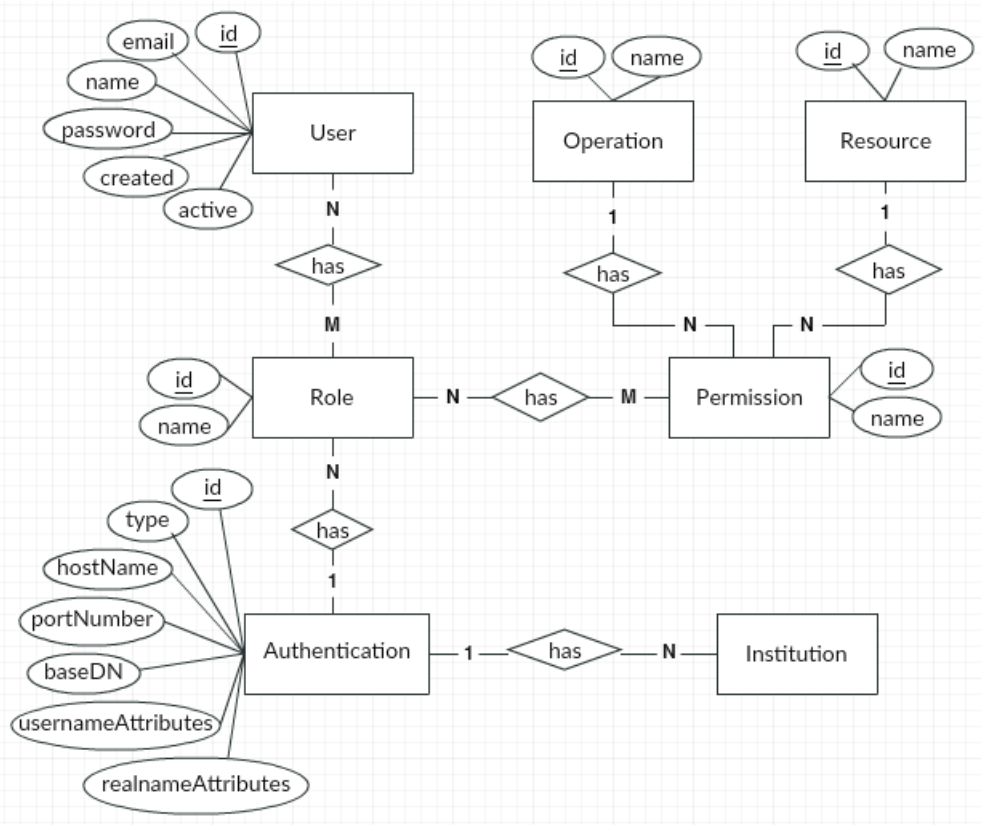



Figure 3.7: RBAC ER model.

### 3.3.3 Overall Architecture

As result of the modeling stage, it was designed an architecture (Figure 3.8) to fit the system requirements. The architecture considers four different tiers: 1) Application tier controls all application functionalities and maintains the system business logic; 2) Data tier is responsible for the maintenance of the private data sources; 3) Processing tier has the duty of executing and monitoring ETL task executions; 4) Client tier is responsible for the solution presentation and page rendering;

The system logic is built upon five different controllers: 1) RBAC controller provides user identity and evaluates access requests to resources; 2) Institution controller allows the creation and destruction of institutions, and resource allocation in institutions; 4) Task controller

enables the creation, configuration and destruction of ETL tasks; 5) Execution controller uses Kettle and Carte servers, so it can build, remotely execute and monitor ETL tasks;

Data Integration (DI) Software Development Kit (SDK) seeks to bridge the gap between Kettle and the stored information in BIcenter database (DB). It provides methods to build Pentaho's ETL processes according to the stored information, and also to execute them. Nevertheless, task execution is a periodic process. Therefore, Execution Scheduler manages the task execution scheduling. When the appropriated time arrives, a Execution Job is triggered. That job will communicate with DI SDK so that the given task is indeed executed. SVG controller maps between images and components, with the intent to build the visual pipeline.

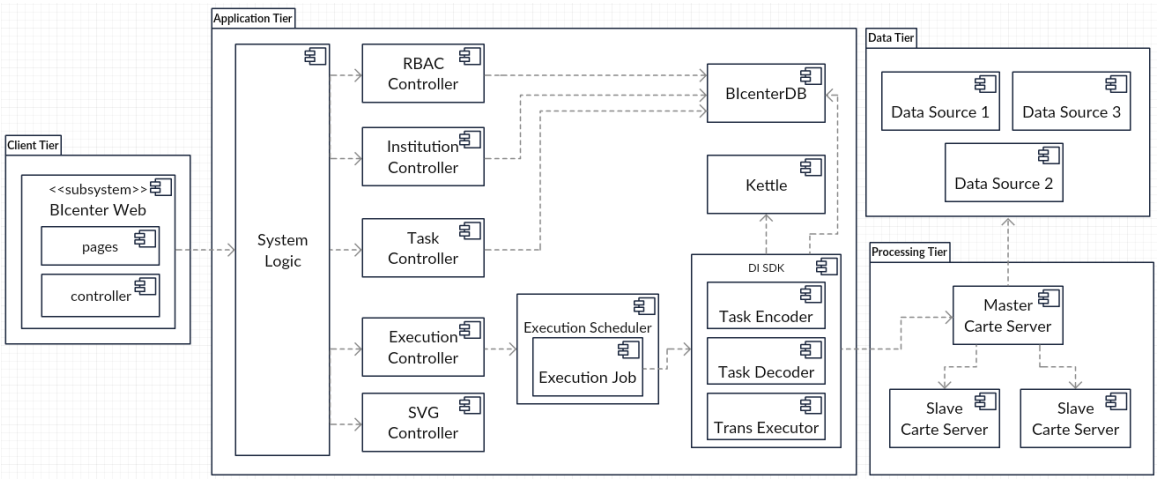



Figure 3.8: Components diagram.

The deployment strategy, represented in Figure 3.9, shows the decoupling between the system and institutions (and their associated resources). Thus, in the system side will be the application server and the system database. In the institution side will be the execution servers and the institution's private data sources. This way, data protection and isolation is guaranteed.

The BIcenter Web client adopted Model-View-Controller (MVC) architectural pattern (Figure 3.10). This pattern divides the web application in three different components:

- **Model:** represents the knowledge and data in an application. It has the responsability to respond to information requests, proceed to information changes according to given instructions requests, and to notify observers in event-driven systems when information changes. Typically, the application data is stored in a database.
- **View:** it represents the user interface. The View updates the UI upon changes in the Model, by rendering the data into the suitable UI form.
- **Controller:** it handles events that occurs in the View, such as user interactions, and updates the Model accordingly.

Moreover, mxGraph [1] is used to draw a visual representation of the ETL tasks stored in

[1] https://jgraph.github.io/mxgraph/

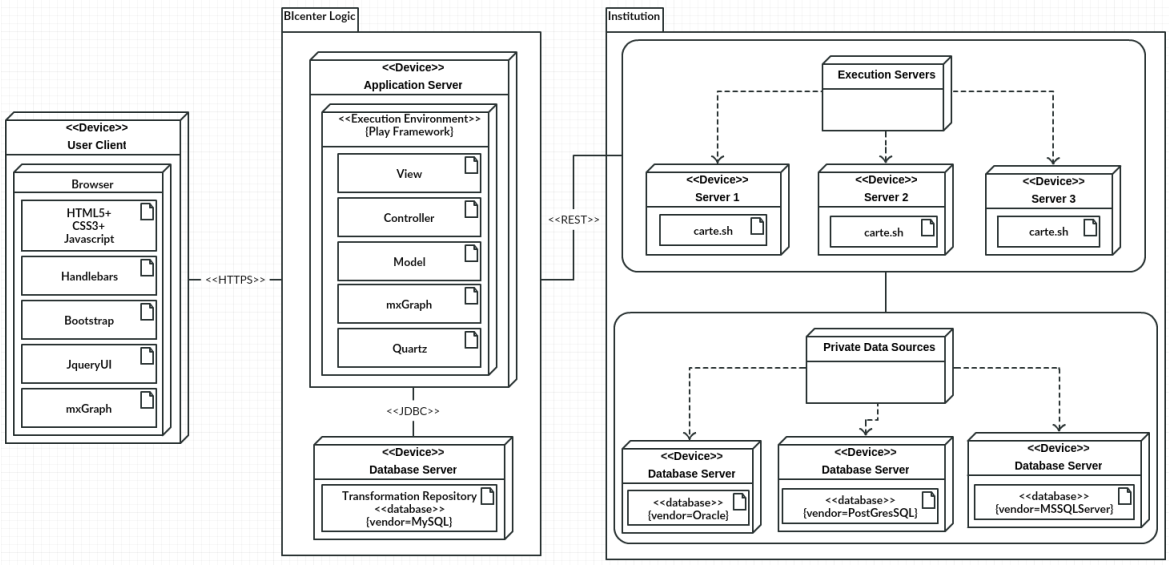



Figure 3.9: Deployment diagram.

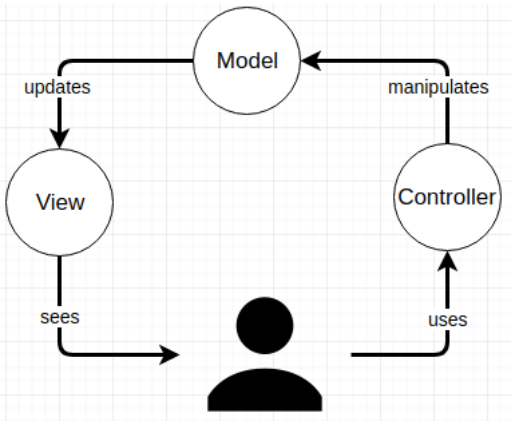



Figure 3.10: MVC architecture.

the system database. Apache Quartz [2] is used to specify scheduled executions of ETL tasks. JQuery UI [3] is used to allow drag and drop actions, so users can build ETL pipelines. The remote execution servers consist of Carte servers. Carte [4] is a simple web server, developed by Pentaho, that allows the remote execution of ETL tasks.

## 3.4 Summary

In this chapter, it was designed a high level solution. The main actors and their stories were identified. As well as the logical overall system architecture and the underlying data model. In the next chapter, it will be described all implementation decisions that materialized the logical architecture.

[2] http://www.quartz-scheduler.org/
[3] https://jqueryui.com/
[4] https://wiki.pentaho.com/display/EAI/Carte+User+Documentation

# Chapter 4

# System Implementation

The choices made during the system implementation were very important to ensure a successful final solution. This chapter explores the basis of the underling system implementation, the problems found during the development phase and the engineered solutions that led to a successful system.

## 4.1 The pipeline editor

So that the user can build a visual representation of the desired ETL pipeline, there is the need to build a proper editor. The editor must be able to process a similar structure to the one depicted in Figure 3.3 and generate the correspondent visual representation of the ETL task. MxGraph [1] was used to build the ETL pipeline editor. MxGraph is a Java/JavaScript diagramming library that enables the building of interactive graphs. A graph consists in a set of cells. A cell can either be a vertex or an hop. So, a graph is formed by a group of vertices connected by edges. The vertices correspond to the ETL steps and the edges correspond to the ETL hops. Figure 4.1 illustrates the MxGraph architecture. Hence, in order to exhibit the desired task within the mxEditor, it is necessary to translate the task to the corresponding mxGraph object. GraphDecoder is responsible for creating a mxGraph and defining the appropriated model, according to the given Task (Algorithm 1). To insert vertices and edges in the graph model, a transaction must be created (beginUpdate and endUpdate). This is required for the model to remain in a consistent state. A default parent is automatically created and represents the first child of the root cell in the model. Subsequent elements must be added to the default parent.

## 4.2 ETL SDK

### 4.2.1 Pentaho Data Integration

Taking into account the evaluations carried out in Section 2.6, it was decided to use PDI SDK. Kettle contains a rich set of data integration functionality that is exposed in a set of data integration tools. However, we can also use Kettle as a library in our own software and solutions. Pentaho Data Integration can be used as a Java API composed by three main components: Core, that contains the core classes for Kettle; Database, that contains

[1] https://jgraph.github.io/mxgraph/

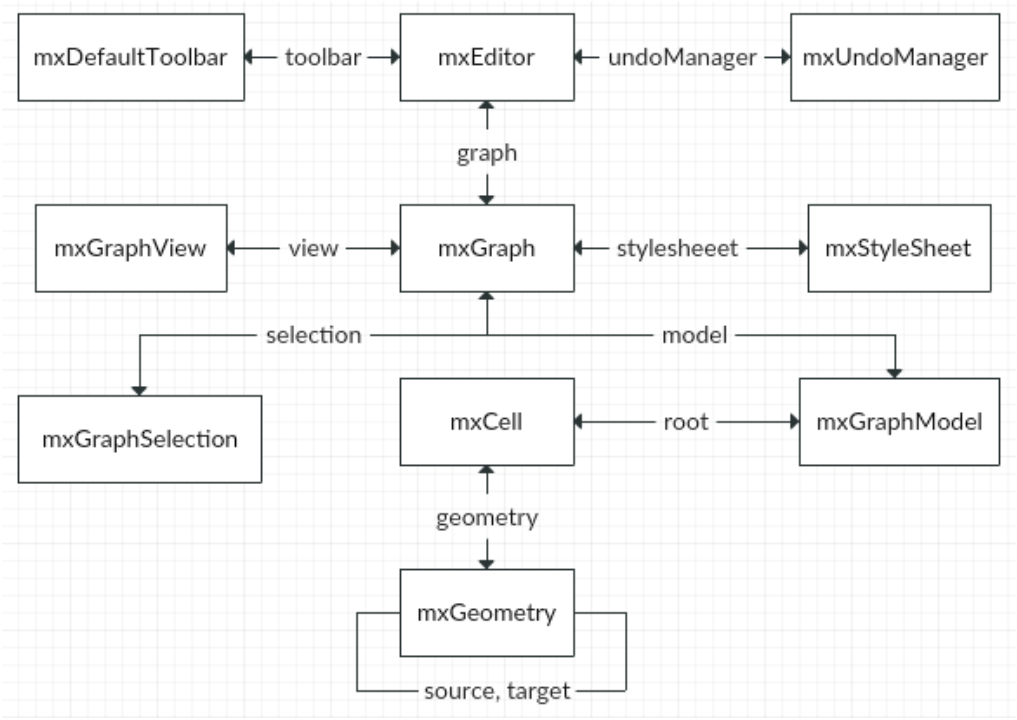



Figure 4.1: MxGraph architecture.

**Algorithm 1**
GraphDecoder algorithm

```
Data: Task
Result: mxGraph
Graph.getModel().beginUpdate()
Parent := Graph.getDefaultParent()
for all Step in Task do
  Id = Step.getId()
  X := Step.Cell.getX()
  Y := Step.Cell.getY()
  Width := Step.Cell.getWidth()
  Height := Step.Cell.getHeight()
  ComponentName := Step.Component.getName()
  StepImage := SvgImageUrl.getUrl(ComponentName, SvgImageUrl.SizeMiddle)
  Cell = Graph.insertVertex(Parent, X, Y, Width, Height, StepImage)
  Cells.put(Id, Cell)
end for
for all Hop in Task do
  SourceId := hop.getSource().getId()
  TargetId := hop.getTarget().getId()
  Graph.insertEdge(Parent,Cells.get(SourceId), Cells.get(TargetId));
end for
Graph.getModel().endUpdate()
```

the database-related classes; Engine, that contains the Kettle's runtime classes. Figure 4.2 explains step by step how an ETL task is executed. The initialization of the Kettle environment loads all available plugins, initializes the logging environment and set up and reads the system variables. After the environment initialization, the transformation metadata is loaded and a transformation engine object is instantiated. Then, the execution is prepared and the transformation threads are started. Finally, because the whole transformation runs multi-threaded, it waits until all processing is completed. The instantiation and configuration of the ETL task object requires an algorithm depicted in Figure 4.3. Nevertheless, each ETL step has its own configuration process and properties.

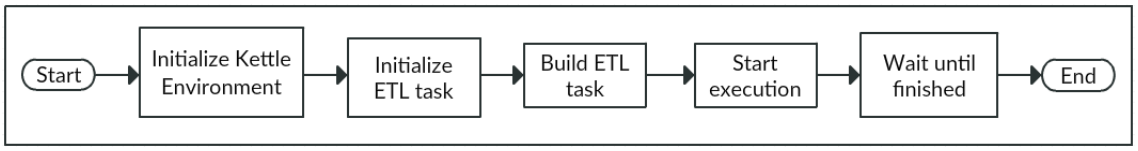



Figure 4.2: ETL task execution flowchart.

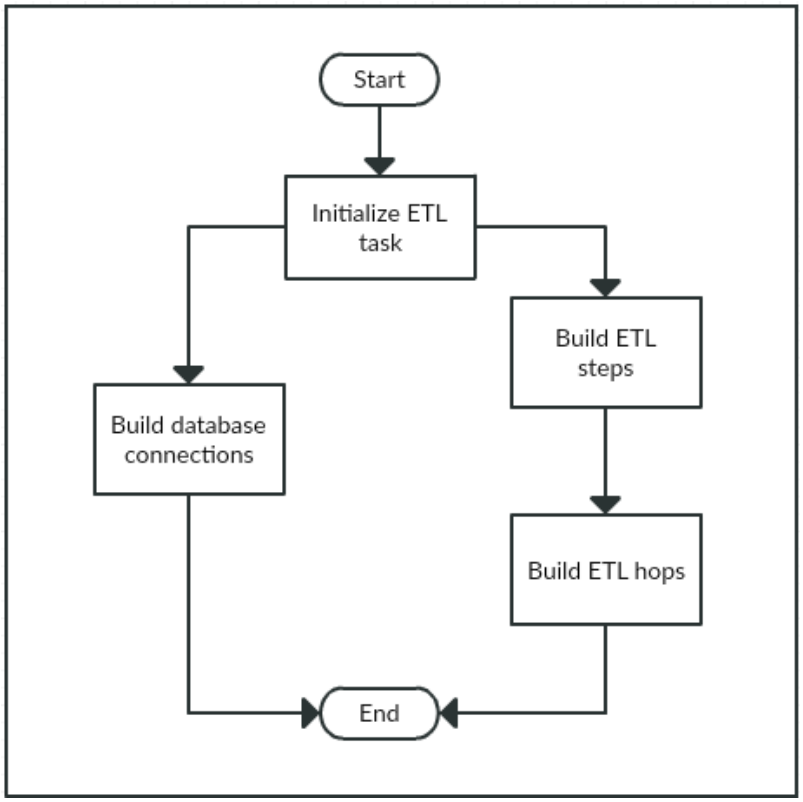



Figure 4.3: ETL task construction flowchart.

### 4.2.2 DI SDK

The database model depicted in Figure 3.4 allows to efficiently store all relevant information in order to build and execute a given ETL process. Thus, one could make use of the stored

data along with Kettle to successfully build and execute ETL tasks. Hence, a DI SDK was designed and developed. In Kettle, an ETL process can be represented by six classes [31]:

- **TransMeta:** is the class that defines the information about the ETL process and offers methods to save and load it from XML, as well as methods to alter an ETL process by adding/removing databases, steps, hops, etc.
- **Trans:** represents the information and operations associated with the concept of an ETL process. It can loads, instantiates, initializes, runs, and monitors the execution of the ETL process.
- **DatabaseMeta:** defines the database specific parameters for a certain database type.
- **StepMeta:** is the class that defines the information about a ETL process's Step.
- **TransHopMeta:** defines a link between two steps in an ETL process.
- **BaseStep:** represents the information and operations associated with the concept of an ETL process Step. It offers initialization, row processing and step clean-up methods.

Thus, initially a TransMeta must be properly configured with all proper DatabaseMetas, StepMetas and TransHopMetas. TransDecoder is responsible to create a TransMeta and define all underlying information, according to the given Task (Algorithm 2). GenericStep is a class that has a generic algorithm to encode or decode a given Step to or from a Kettle's StepMeta. In order to provide interface segregation, TransDecoder must use the StepDecoder interface that only provides the GenericStep decoding method (Figure 4.4).

**Algorithm 2**
TransDecoder algorithm

```
Data: Task
Result: TransMeta
for all DataSource in Task do
  TransMeta.addOrReplaceDatabase(DataSource);
end for
for all Step in Task do
  StepMeta := StepDecoder.decode(Step)
  TransMeta.addStep(StepMeta)
end for
for all Hop in Task do
  TransHopMeta.setFromStep(Hop.source)
  TransHopMeta.setToStep(Hop.target)
  TransMeta.addTransHop(TransHopMeta)
end for
```

After having a correctly configured TransMeta, it can be executed with the associated Trans object. TransExecutor is a singleton responsible for the initialization, execution and monitoring of the Trans object. Moreover, it writes and stores the execution logs, step measures and status, and the resulting data in real-time. To accomplish this, listeners are coupled to the Trans and Steps objects in order to obtain the results and metrics throughout the transformation's execution (Figure 4.5).

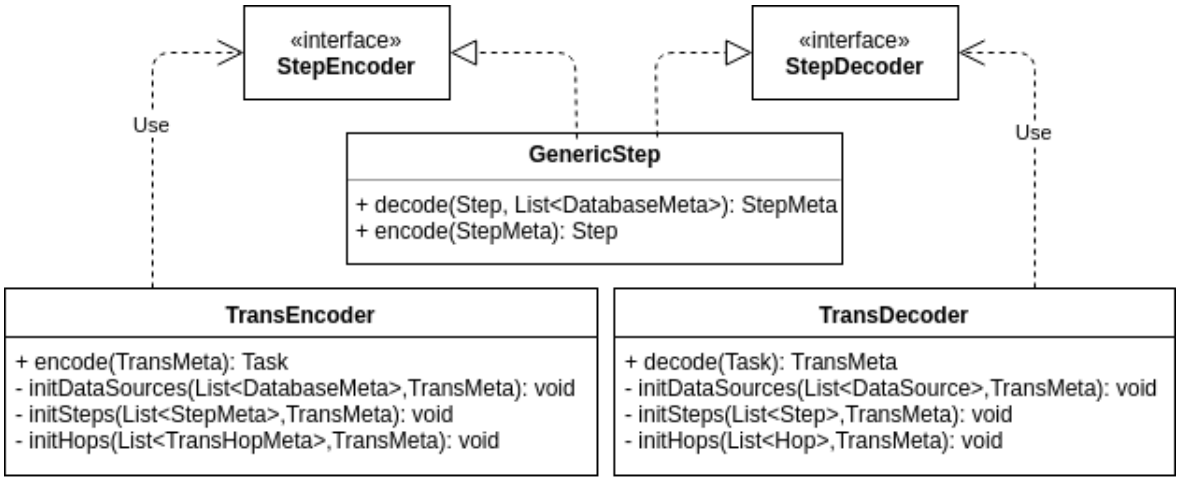



Figure 4.4: Trans builder class diagram.

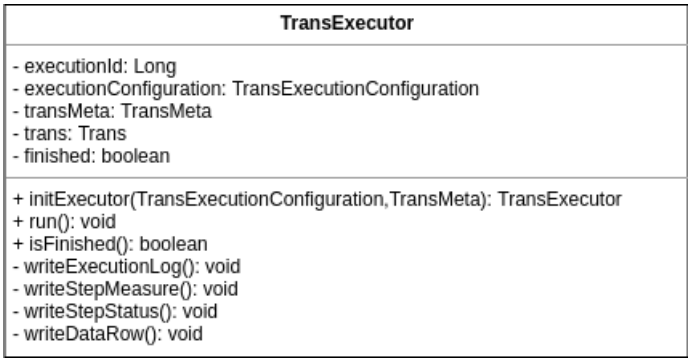



Figure 4.5: TransExecutor class.

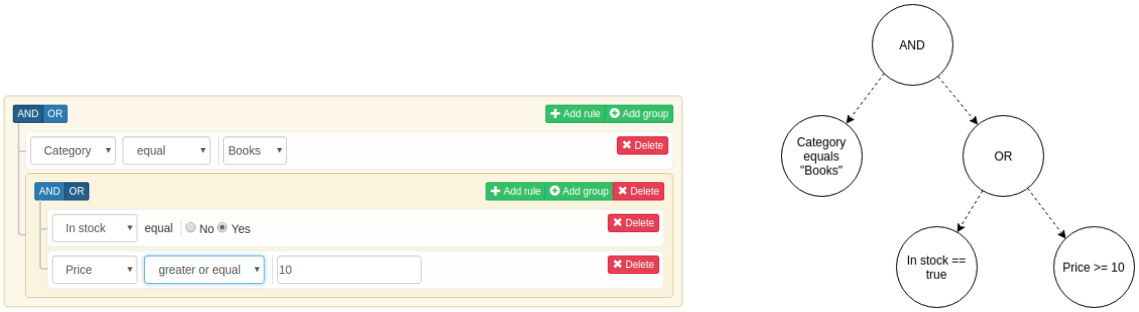
### 4.2.3 Rendering algorithm

As stated previously, the effort of developing new components is a repetitive work. Therefore, GenericStep has a generic algorithm to dynamically encode/decode a Kettle Step. As shown by Algorithm 3, it is used a reflective software approach to traverse over the StepProperties. For each StepProperty, the correspondent StepMetaInterface method will be invoked with the stored value. Nonetheless, since StepProperty values has different types and formats, and considering that within the database the value is stored as a String, the invokeMethod function must dynamically cast the value to the desired type. Thus, invokeMethod uses reflection to check the method parameter and dynamically cast the given value to the desired type. If a given StepProperty represents a json table, for each associated ComponentMetadata, the correspondent value will be extracted from the json, and the invokeMethod will be executed accordingly.

**Algorithm 3**
GenericStep decode algorithm

```
Data: Step
Result: StepMeta
StepMetaInterface = PluginRegistry.findPlugin(Step.name)
for all StepProperty in Step do
  if StepProperty.ComponentProperty.type != table then
    for all Method in StepMetaInterface's setMethods do
      if StepProperty.ComponentProperty.method == Method.name then
        invokeMethod(StepMetaInterface, Method, StepProperty.value)
      end if
    end for
  else
    for all ComponentMetadata in StepProperty.ComponentProperty do
      for all Method in StepMetaInterface's setMethods do
        if ComponentMetadata.method == Method.name then
          invokeMethod(StepMetaInterface, Method, StepProperty.value
          .getComponentMetadataValue(ComponentMetadata.id))
        end if
      end for
    end for
  end if
end for
```

Some input parameter are complex and require a casting algorithm. Several PDI components, for instance FilterRows (which role is to filter data rows based on a given boolean condition), need a Condition object. There are two types of conditions: atomic conditions and composite conditions. An atomic condition is defined by a field, a function and a value. A composite condition is composed by a set of atomic or composite conditions linked by logical operators (NOT Condition1 AND Condition2 OR Condition3 ). The boolean condition is defined on the web interface (Figure 4.6a) and stored in the database as a Json String (Figure 4.6b). Thus, Algorithm 4 is a recursive algorithm that traverses the tree structure and builds the Condition object accordingly.




(a) Web interface (b) Tree structure

Figure 4.6: Boolean condition

**Algorithm 4**
ConditionBuilder

```
Data: Operator, Json
Result: Condition
if Json has no children then
  Condition.setLeftValuename(Json.field)
  Condition.setFunction(Json.function)
  Condition.setRightExact(value)
else
  for all Child in Json do
    if Child is not last then
      SubCondition := ConditionBuilder(Json.operator,Child);
    else
      SubCondition := ConditionBuilder("NONE",Child);
    end if
  end forCondition.addCondition(SubCondition);
end if
if Json.operator != "None" then
  Condition.setOperator(Json.operator);
end if
```

### 4.2.4 Extensibility

The definition of the ETL components is maintained in a JSON file which is automatically processed during the application startup. In this phase, all Component objects are extracted and stored within the platform's database. When the user wants to configure a step or wants to run an ETL process, the application loads the needed Components, in order to build the settings page or to create the StepMeta object. One can describe new ETL components in the configuration file. The Step will be immediately supported by the web platform because of the dynamic and reflection-based approach. For instance, SortRows can be specified as presented in Figure 4.7.


```
{
   "name":"SortRows",
   "label":"Sort Rows",
   "componentProperties":[
      {
         "label":"Step Name",
         "shortName":"setStepName",
         "type":"input"
      },
      {
         "label":"Fields",
         "type":"table",
         "componentMetadatas":[
            {
               "label":"Field Name",
               "method":"setFieldName",
               "type":"select",
               "source":"inputFields"
            },
            {
               "label":"Ascending",
               "method":"setAscending"
            },
            {
               "label":"Case Sensitive Compare?",
               "method":"setCaseSensitive"
            }
         ]
      }
   ]
}
```


Figure 4.7: SortRows component specification.

## 4.3 Execution servers

In the architecture design, it was defined the need of dedicated execution servers allocated to institutions, so private data protection and isolation is guaranteed. Carte is a lightweight HTTP server that allows remote and parallel execution of ETL tasks. It runs in background and listens for requests to execute specific transformations. While Carte is running, it waits and listens for requests on a predefined network port (TCP/IP). Clients on a remote computer

can make requests to a machine running Carte, sending an ETL task definition as part of the request message. When a running Carte instance receives a given request, it authenticates the request and executes the transformation contained in it.

Clustering allows a single transformation to be divided and executed in parallel by multiple computers that are running the Carte server, thus distributing the workload. It accepts commands from remote clients. These commands control the deployment, management, and monitoring of transformations on the slave server.

A slave server is the smallest building block of a cluster. It can be started by specifying the hostname or IP address and the port the Web server should run on. Kettle relies on a XML format for the configuration of the slave server (Figure 4.8). The <slaveserver> XML block describes the hostname and port that the slave server should listen to, and it allows the configuration of various aspects of the slave server. In general, these options enable to perform a fine-tune of the memory usage for a long-running server process such as Carte:

- **max_logs_lines:** allows to configure the maximum number of log lines that the slave server logging system should keep in memory at most.
- **max_logs_lines:** describes the maximum time in minutes that a log line should be kept in memory. This is especially useful for long-lived transformations to prevent the slave server from running out of memory.
- **object_timeout_minutes:** By default, all transformations stay visible in the slave server status report indefinitely. This parameter allows to automatically clean out old transformations from the status list.

```
<slave_config>
    <!-- Simple slave server configuration -->
    <max_log_lines>0</max_log_lines>
    <max_log_timeout_minutes>0</max_log_timeout_minutes>
    <object_timeout_minutes>5</object_timeout_minutes>

    <slaveserver>
        <name>server1</name>
        <hostname>server1</hostname>
        <port>8181</port>
    </slaveserver>
</slave_config>
```

Figure 4.8: Slave server configuration.

A cluster schema consists of one master server that controls the cluster, and a number of non-master slave servers. It contains metadata on how master and slaves must transmit data. Data is passed between Carte servers over TCP/IP sockets. When a transformation is running, the different parts of it are distributed across Carte slave server nodes for processing, while the Carte master server node tracks the progress. To set up a dynamic cluster, initially a single Carte server instance must be launched to work as the master node (Figure 4.9a). Since slaves need to be able to contact the master node, it usually has a fixed IP address or hostname. This configuration can be supplemented with additional logging parameters like Figure 4.8. Hereafter, Carte slaves can register to the previously launched Carte master

(Figure 4.9b). In the <masters> section of the slave server's configuration file, one or more Carte servers can be specified to which the slave needs to report.

```
<slave_config>
    <slaveserver>
        <name>Master</name>
        <hostname>yourhostname</hostname>
        <port>9001</port>
        <username>cluster</username>
        <password>cluster</password>
        <master>Y</master>
    </slaveserver>
</slave_config>
```

(a) Carte master

```
<slave_config>
    <masters>
        <slaveserver>
            <name>Master</name>
            <hostname>yourhostname</hostname>
            <port>9000</port>
            <username>cluster</username>
            <password>cluster</password>
            <master>Y</master>
        </slaveserver>
    </masters>

    <report_to_masters>Y</report_to_masters>

    <slaveserver>
        <name>SlaveOne</name>
        <hostname>yourhostname</hostname>
        <port>9001</port>
        <username>cluster</username>
        <password>cluster</password>
        <master>N</master>
    </slaveserver>
</slave_config>
```

(b) Carte slave

Figure 4.9: Dynamic Carte cluster configuration

## 4.4 Task scheduler

Considering the importance of executing ETL tasks in a scheduled and periodic manner, it is important to build a component that manages and triggers executions. When one wants to schedule a task, a start date and a periodic interval must be defined. Therefore, a execution job scheduler was developed using Apache Quartz [2]. Quartz is a Java job scheduling library that allows the creation of simple or complex schedules for executing jobs. To schedule a job execution, Quartz allows the definition of a CronTrigger. The CronTrigger class is based on the scheduling capabilities of cron. A cron expression is a string comprising five or six fields that represents a schedule to execute some routine. Table 4.1 defines all cron expression fields. Thus, a cron expression string can be built based on the previously announced fields.

| Field Name | Mandatory | Allowed Values | Allowed Special Characters |
|---|---|---|---|
| Seconds | Yes | 0-59 | ,-*/ |
| Minutes | Yes | 0-59 | ,-*/ |
| Hours | Yes | 0-23 | ,-*/ |
| Day of month | Yes | 1-31 | ,-*?/LW |
| Month | Yes | 1-12 or JAN-DEC | ,-*/ |
| Day of week | Yes | 1-7 or SUN-SAT | ,-*?/L# |
| Year | No | empty, 1970-2099 | ,-*/ |

Table 4.1: Cron expression fields.

[2] http://www.quartz-scheduler.org/

## 4.5 Summary

In this chapter, it was explained the main development challenges and the subsequent engineered solutions. Firstly, it was described the translation process between the data model and the visual representation of an ETL task. Also, it was described the translation process from the data model to the Pentaho's transformation object, so one can execute the desired task. It was presented the algorithm that allows a dynamic integration of ETL components within the platform. Furthermore, it was explained how an execution server can be deployed and executed. Finally, it was explained how one can periodically schedule ETL tasks. In the next chapter, the final platform will be explained step by step and a use case will be presented.

# Chapter 5

# BICenter: Results and Use cases

The objectives defined on the system modeling phase were successfully fulfilled. As a result, a system with a large potential to facilitate the management of ETL processes in a multi-institution environment was achieved.

## 5.1 BIcenter User Interface

In this section, BIcenter UI and all underlying interactions will be presented in detail. In the login page (Figure 5.1), users can be authenticated using an institution LDAP, which provides a central place to store usernames and passwords.

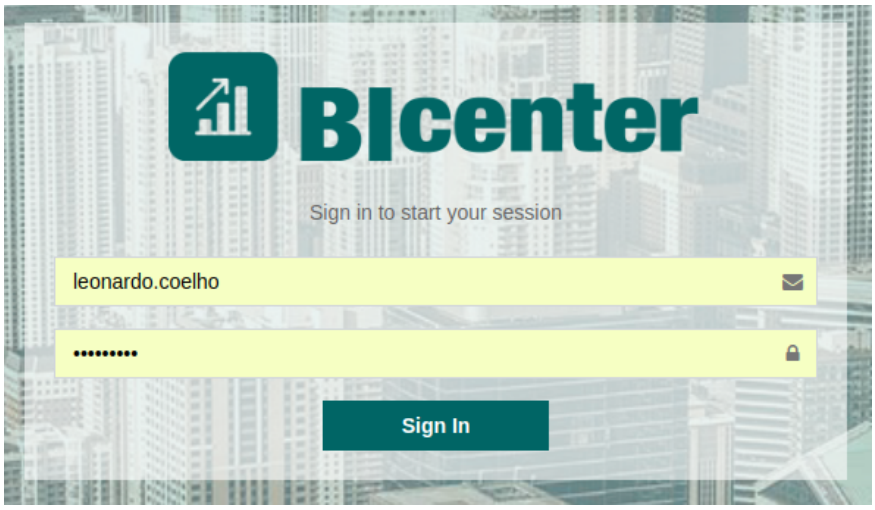



Figure 5.1: Login page.

After user authentication, he is redirected to the application's main page (Figure 5.2). The main page offers an ETL pipeline editor. Users can drag components located on the sidebar and drop them on the editor. Also, it provides a toolbar (Figure 5.3) that offers edition and selection functionalities over the ETL pipeline. Moreover, it allows to configure and inspect all pipeline steps and to execute the ETL task.

In the sidebar, one can list all accessible institutions. Each institution is composed by an execution scheduler, a task folder, a server folder and a data source folder (Figure 5.4). After

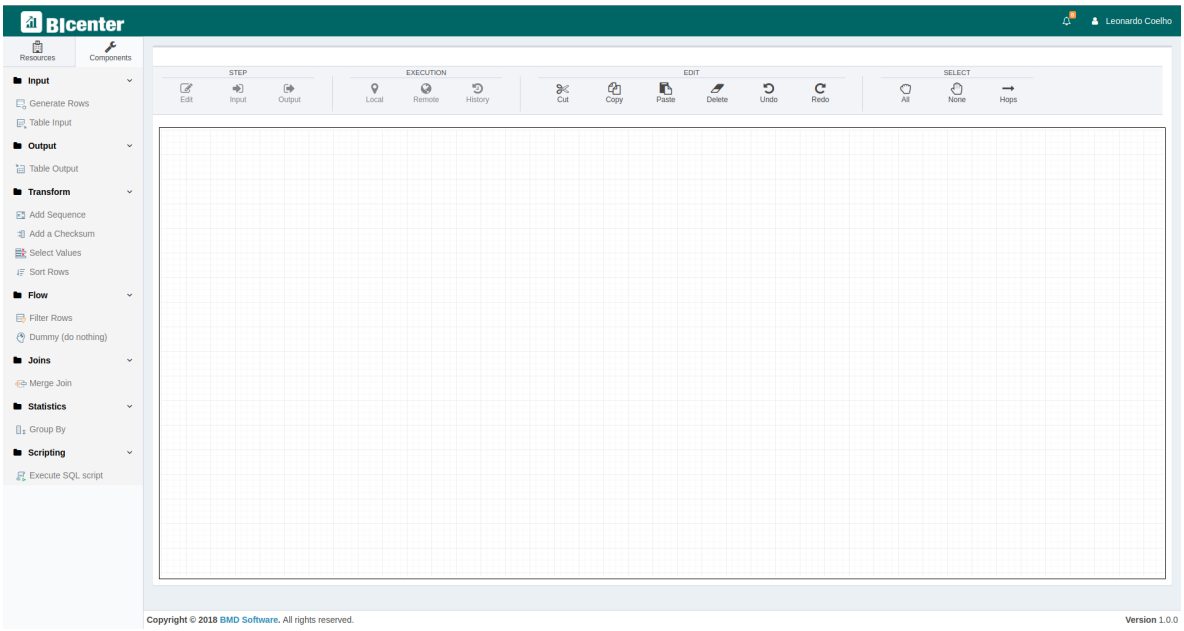



Figure 5.2: Main page.


Figure 5.3: Editor toolbar.

opening the task folder, a new task can be created by specifying its name. Thereafter, the ETL task can be built by dragging components and connecting them together.

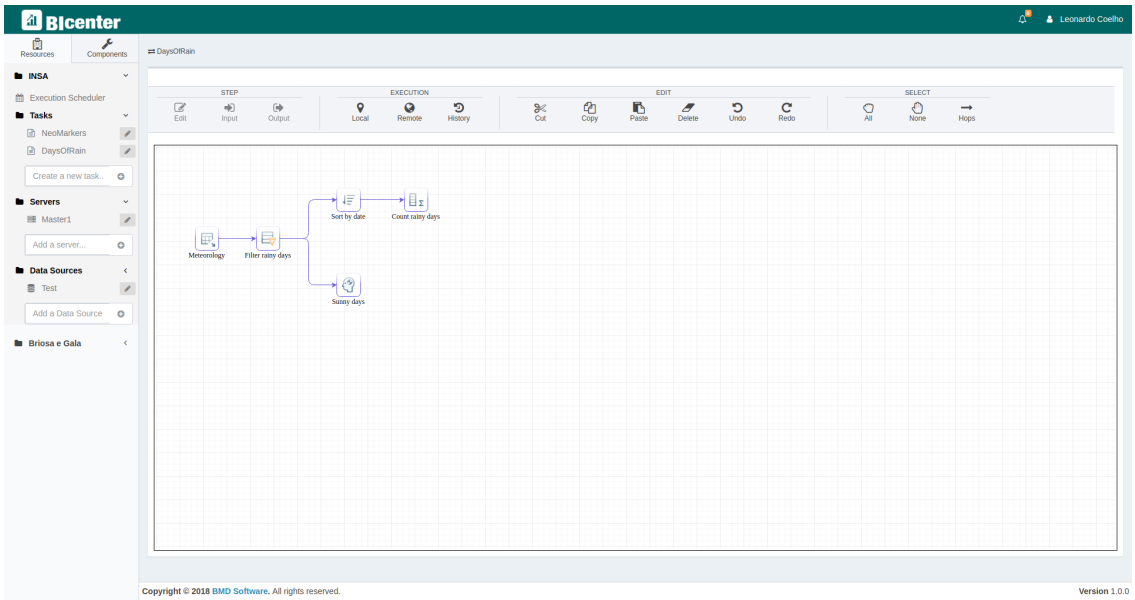



Figure 5.4: Institution page.

In Figure 5.5 represents an example of an ETL task. This task is able to process the rainfall dataset from a weather station, filtering all rainy days, and counting the number of rainy days by month and year. After drawing the ETL task, all data sources, components and the execution server must be properly configured.

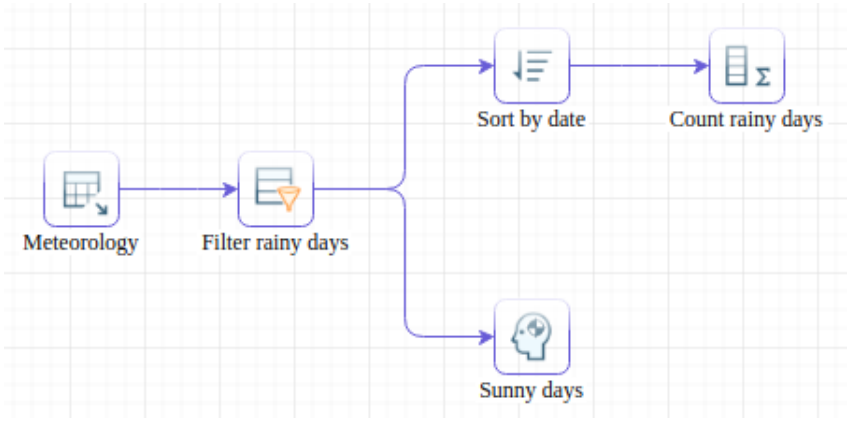
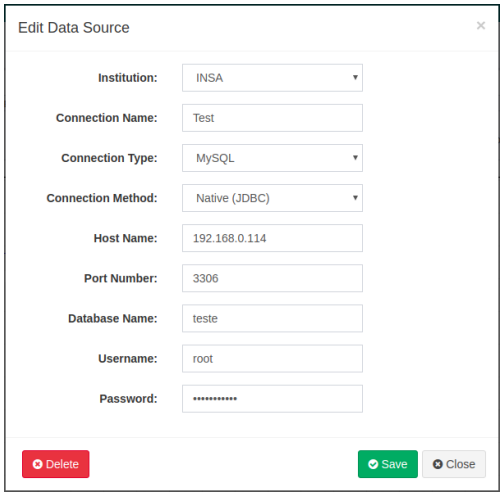
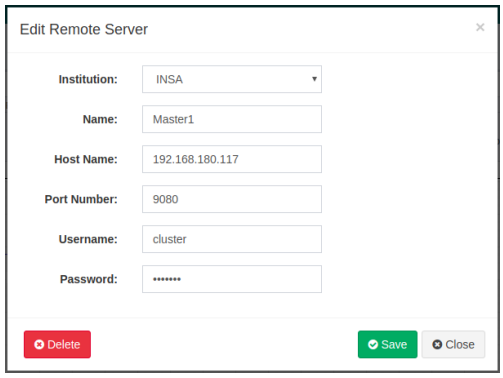



Figure 5.5: Count rainy days by month.

To configure a data source, one can open the data source folder and, thus, a popup will arises (Figure 5.6a). That popup allows to describe the data source definition. Simlarly, to configure a remote execution server, one can open the server folder and describe all server properties within the corresponding popup (Figure 5.6b).




(a) Data source settings page.




(b) Execution server settings page.

Figure 5.6: Resource settings

After configuring the data sources and the execution server, each ETL component must be configured through the web interface. For instance, Figure 5.7 shows the Filter Rows configuration page. In order to configure the component, it must be specified a boolean condition that allows to filter the input rows. After applying the condition to a given row, it must be fowarded to a certain step. Thus, it should also be defined the ETL steps for the positive and negative case.

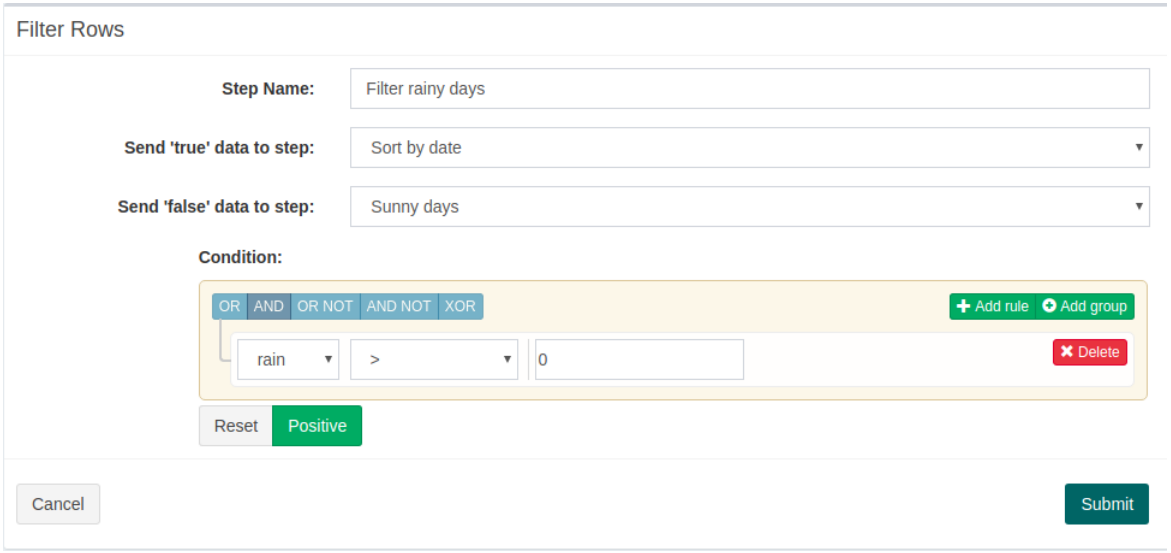



Figure 5.7: Filter rows settings page.

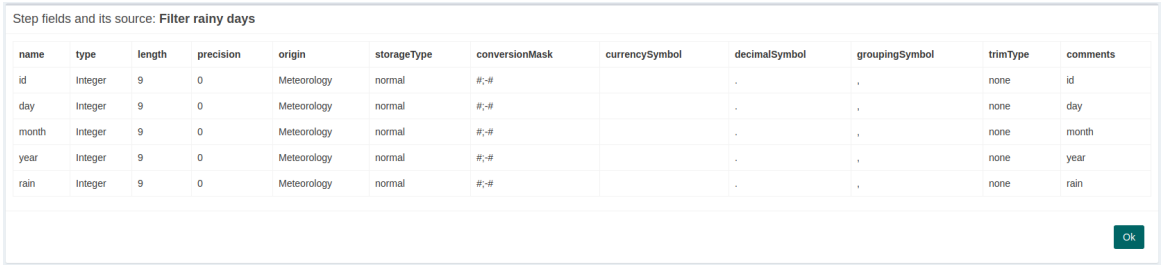



Figure 5.8: Filter rows input fields.

When all steps are configured, the ETL task can be executed. As can be seen on Figure 5.3, the task may run locally or remotelly. Preferably, the task should run on the remote server, that belongs to the institution, to ensure data protection and isolation. When it is a remote execution, it can be scheduled. If the remote toolbar execution button is clicked, a popup will appear allowing the scheduling of the ETL task (Figure 5.9). Firstly, the remote execution server must be chosen. Secondly, if the scheduling of the task is desired, a execution date must be defined. Finally, to execute the task periodically, an interval must be specified.

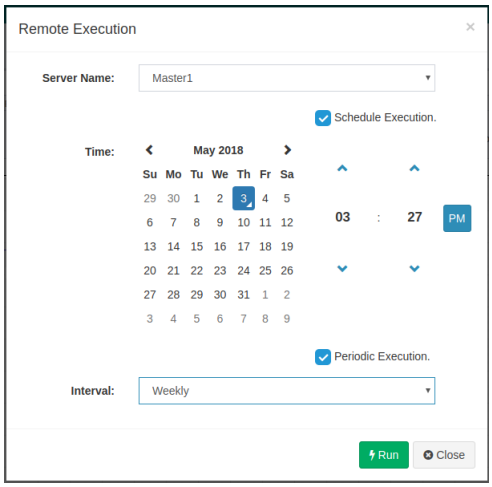



Figure 5.9: Scheduling a ETL task on a remote execution server.

On the sidebar, when "Execution Scheduler" is clicked, it redirects to the execution scheduling table (Figure 5.10). Here one can check or delete previously defined schedules.

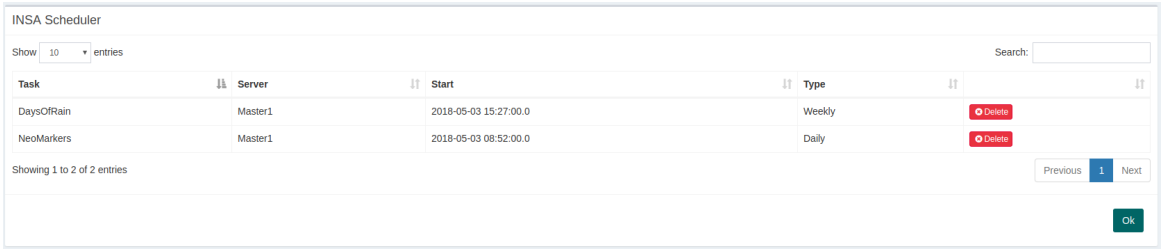



Figure 5.10: Execution scheduling table.

When some execution finishes, the user will be notified by the navbar's bell (Figure 5.2). The execution history can be analyzed at any time (Figure 5.11). The history provides a timeline where one can see all executions' start date and time, the used execution server, the duration time and the user who have defined the given schedule.

For each execution, users can verify the execution logs (Figure 5.12), steps' performance metrics (Figure 5.13) and preview output results (Figure 5.14) on real-time. The real-time output data can be displayed for each step (Figure 5.14b).

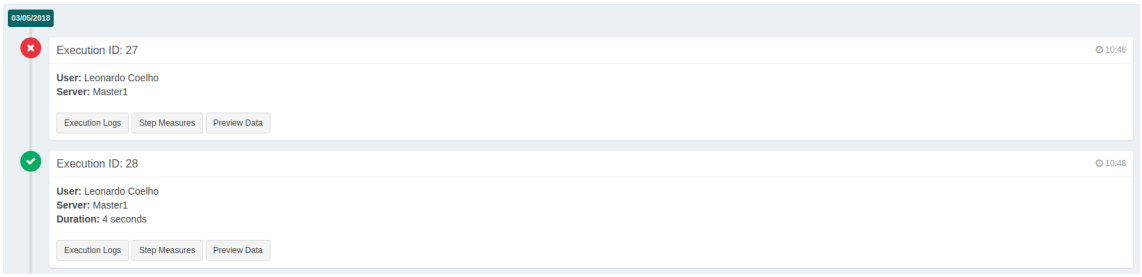



Figure 5.11: Execution history.


Figure 5.12: Execution logs.

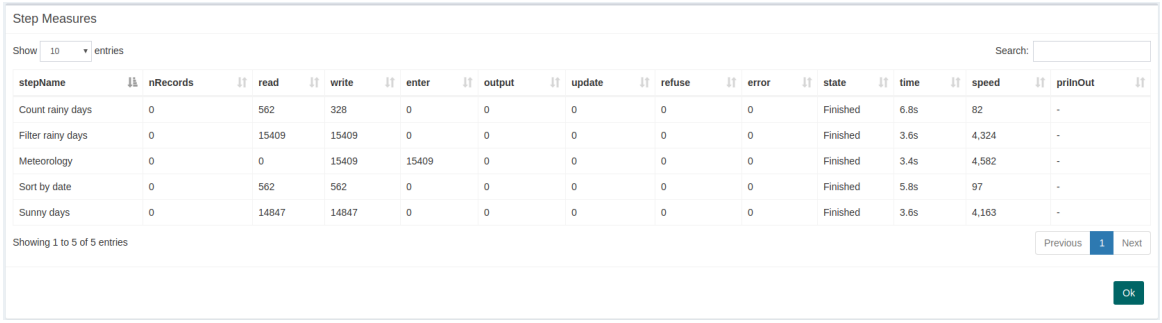
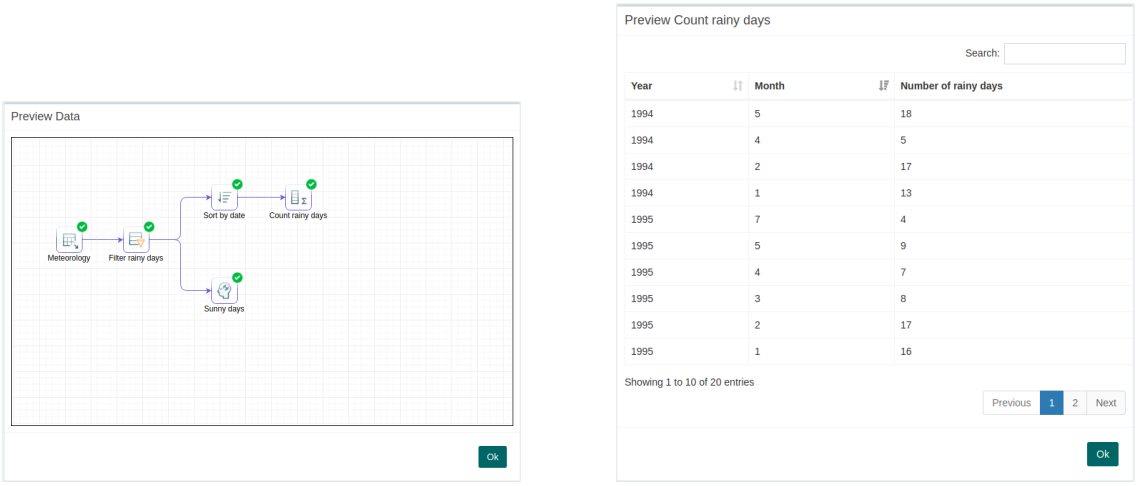



Figure 5.13: Execution step measures.




(a) Steps status

(b) Preview step output

Figure 5.14: Preview execution output.

## 5.2 A use case on biomarkers analysis

Mass spectrometry is a physical analytical technique to detect and identify molecules of interest by measuring its mass and characterizing its chemical structure. In order to identify the relevant molecules, a set of biomarkers are analyzed and evaluated. Therefore, a set of test tubes are placed in a plate and are applied for screening. To fine tune the data machine, periodically, biomarkers mean values are calculated.

Figure 5.15 illustrates the ETL task that calculates the biomarkers averages'. Firstly, plates are filtered by date. Then, plates, screens and screening values are merged. Hence, the values of the various biomarkers are averaged. Finally, those averages are written to a database table (Figure 5.16). This pipeline was performed on a newborn dataset. Due to data sensitivity, only a general overview is presented.

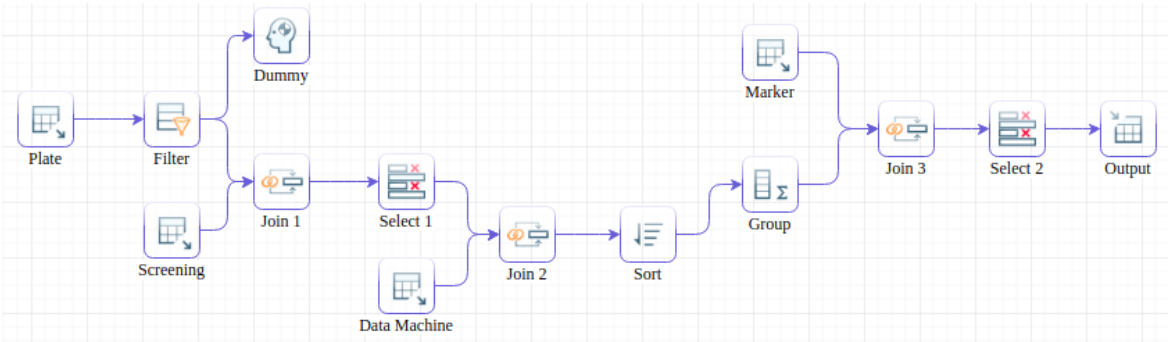



Figure 5.15: ETL task to calculate biomarkers averages.

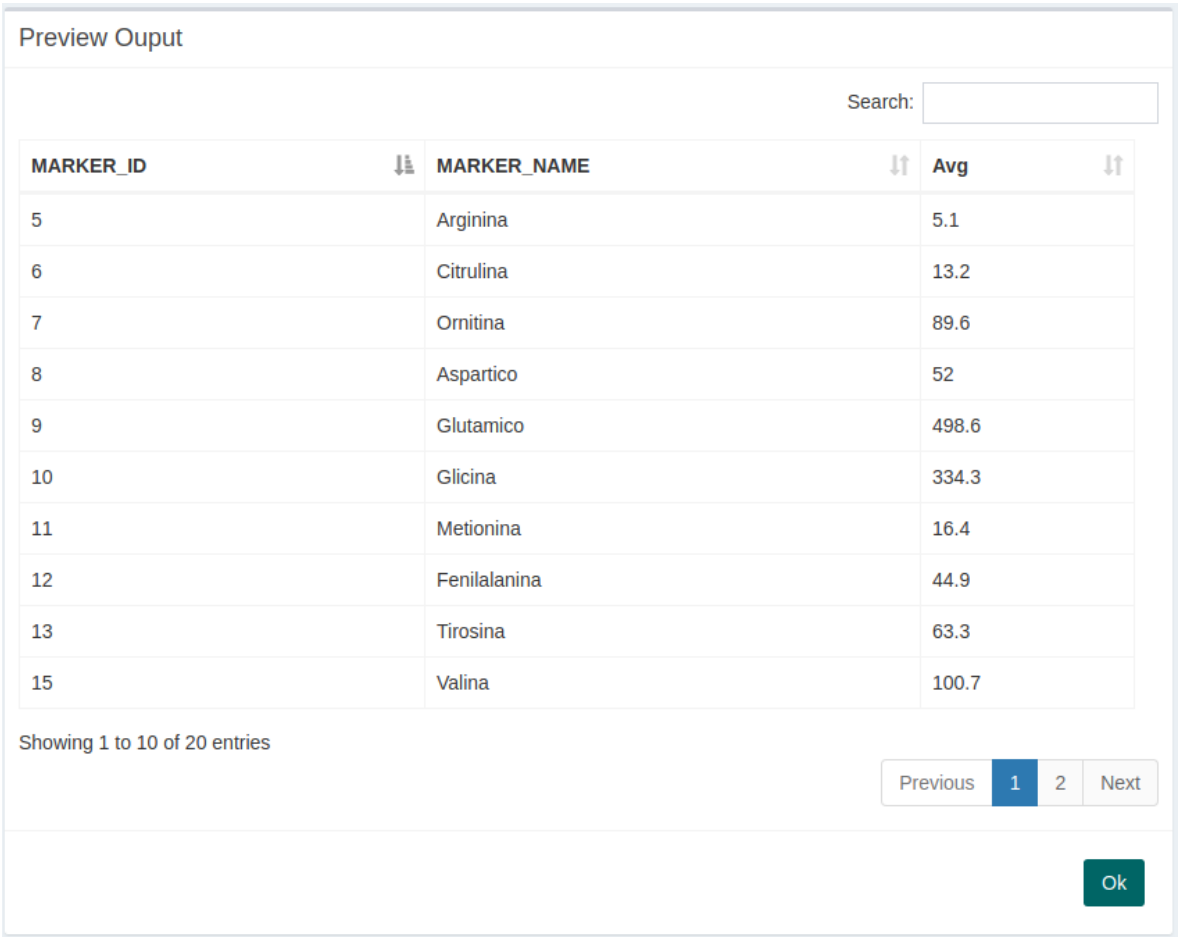



Figure 5.16: Biomarkers averages.

# Chapter 6

# Conclusion

The main objective of this thesis was the study, design and development of a solution that covers the limitations and problems found in the building and management of ETL tasks in a multi-institution environment. From the initial study, it was concluded that a web solution would translate into a significant positive impact for ETL task developers, since it would facilitate the installation and maintenance process. Moreover, considering that ETL tasks are usually related with sensitive data, it was important to devise an architecture that would garantees data protection and isolation. Since ETL tasks are tipically periodically executed, for example to feed a data warehouse or to produce statistical reports, it was also necessary to allow execution's scheduling. Finally, there was the need for user authentication and access control to institutions and resources. Thus, a RBAC system was developed, and it can connects to institutions' LDAP or AD. Thus, users that belongs to a certain institution don't need to register in the platform, since they already have access to that institution.

To implement the system that achieved these goals, an architecture that identified the main key logical components was designed and a study of the appropiate technologies for each component was carried out. Therefore, mxGraph was used to translate and display a visual representation of the stored ETL tasks. Kettle was used to allow the dynamic definition of transformations based on the stored ETL tasks. In addition, it allowed one to use all existing Pentaho ETL components. Nonetheless, Kettle allows the implementation of new components as a custom plugin. Carte servers enabled to execute transformations remotely. Those servers will reside in institutions, as well as the private data sources, thus ensuring data protection. Additionally, Carte servers can be defined as a cluster formed by a master and a group of slaves. This cluster configuration allows load balancing and failover operations. Apache Quartz was used to allow the scheduling of ETL task executions through cron expressions.

All system requirements listed in chapter 3 were fulfilled. The final system allows users, if they have the right permissions, to instantly start developing and scheduling ETL tasks and to define execution servers and private data sources within a institution. Also, the integration effort of new ETL components was mitigated by the dynamic approach to component specification.

## 6.1 Future Work

Despite the obtained results, several developments can be planned as future work. Considering that one of the most important use cases for ETL tasks is to aggregate data in order

to produce statistical reports, a report management tool would be an important contribution. Anyway, one has to think and decide if the report management tool should be integrated in BIcenter or if it should be a standalone product, which would interact in a coordinated way with BIcenter. Since currently the ETL step ouput is only represented in a tabular format, there is a limitation on the value that can be extracted from the data during the ETL task execution. Hence, it would be extremely important to provide ways to define enriched views of the output, namely pivot tables, bar charts, line charts, area charts, scatter charts, pie charts and heat grids. There are several data visualization libraries that could be good options to empower enriched views over the ouput data, for example: Pentaho Visualization API, D3.js and Chartist.js.

# Bibliography


[1] T. M. C. et al, "Opportunities and challenges of big data in economics research and enterprises," *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 4, pp. 1155–1161, 2014.

[2] J. V. G. Matt Casters, Roland Bouman, *Pentaho Kettle Solutions*. Wiley Publishing, Inc., 2010.

[3] A. Labrinidis and H. V. Jagadish, "Challenges and opportunities with big data," *Proc. VLDB Endow.*, vol. 5, pp. 2032–2033, Aug. 2012.

[4] P. P. L. R. Ruchita H.Bajaj, "Big data – the new era of data," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 2, pp. 1875–1885, 2014.

[5] W. K. Motashim Rasool, "Big data: Study in structured and unstructured data," *HCTL Open International Journal of Technology Innovations and Research (IJTIR)*, vol. 14, pp. 1–6, 2015.

[6] P. B. Divyakant Agrawal, "Challenges and opportunities with big data," *Cyber Center Technical Reports*, 2012.

[7] S. Negash, "Business intelligence," *Communications of the Association for Information Systems (CAIS)*, vol. 13, no. 15, pp. 177–196, 2004.

[8] Margaret Rouse, "business intelligence (bi)." `https://searchbusinessanalytics.techtarget.com/definition/business-intelligence-BI`, 2016. [Online; accessed 2018-May-21].

[9] H. W. Inmon, *Building the Data Warehouse*. New York: John Wiley & Sons Inc, 3 ed., 2002.

[10] E. Rahm and H. Hai Do, "Data cleaning: Problems and current approaches," *IEEE Computer Society*, vol. 23, no. 4, pp. 3–13, 2000.

[11] N. Mali, "A Survey of ETL Tools," *International Journal of Computer Techniques -–*, vol. 2, no. 5.

[12] P. S. Metkewar, "A Technical Comprehensive Survey of ETL Tools," *International Journal of Applied Engineering Research ISSN*, vol. 11, no. 4, pp. 973–4562, 2016.

[13] R. Krishna and Sreekanth, "An object oriented modeling and implementation of web based etl process," *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 2, pp. 192–196, 2010.

[14] V. Radhakrishna, V. Sreekanth, and B. RangaSwamy, "Implementation ofweb-etl transformation with pre-configured multi-source system connection and transformation mapping statistics report," *International Conference on Advanced Computer Theory and Engineering(ICACTE)*, vol. 3, no. 2, pp. 317–322, 2010.

[15] M. Novak and K. Rabuzin, "Prototype of a web etl tool," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 5, no. 6, pp. 97–103, 2014.

[16] S. N. et al, "An improved version of big data classification and clustering using graph search technique," *International Journal of Computer Science and Mobile Computing*, 2016.

[17] P. D. Juarez, P. Matthews-Juarez, D. B. Hood, W. Im, R. S. Levine, B. J. Kilbourne, M. A. Langston, M. Z. Al-Hamdan, W. L. Crosson, M. G. Estes, S. M. Estes, V. K. Agboto, P. Robinson, S. Wilson, and M. Y. Lichtveld, "The public health exposome: A population-based, exposure science approach to health disparities research," *International Journal of Environmental Research and Public Health*, vol. 11, no. 12, pp. 12866–12895, 2014.

[18] Wikipedia, "Data warehouse." `https://en.wikipedia.org/wiki/Data_warehouse`, 2017. [Online; accessed 2018-May-21].

[19] Margaret Rouse, "Data mart." `https://searchsqlserver.techtarget.com/definition/data-mart`, 2014. [Online; accessed 2018-May-21].

[20] Margaret Rouse, "Star schema." `https://searchdatamanagement.techtarget.com/definition/star-schema`, 20010. [Online; accessed 2018-May-28].

[21] Margaret Rouse, "Olap (online analytical processing)." `https://searchdatamanagement.techtarget.com/definition/OLAP`, 2007. [Online; accessed 2018-May-28].

[22] M. A. Beyer, E. Thoo, E. Zaidi, and R. Greenwald, "Magic quadrant for data integration tools," tech. rep., CloverETL, Gartner, 08 2016.

[23] H. Filho, "Análise comparativa das ferramentas de ETL - Kettle e Talend," Master's thesis, UNIVERSIDADE FEDERAL DA PARAÍBA, RIO TINTO – PB, 2013.

[24] T. A. Majchrzak, T. Jansen, and H. Kuchen, "Efficiency evaluation of open source etl tools," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, (New York, NY, USA), pp. 287–294, ACM, 2011.

[25] M. Russel, "Etl benchmarks," tech. rep., Manapps, info@manapps.tm.fr, 10 2008.

[26] R. Katragadda, S. Sremath Tirumala, and D. Nandigam, "ETL tools for Data Warehousing: An empirical study of Open Source Talend Studio versus Microsoft SSIS,"

[27] H. S. Rose and P. Raibagkar, "A Comparative Study of ETL Tools," *IJSRD -International Journal for Scientific Research & Development|*, vol. 4, no. 05online, pp. 2321–613, 2016.

[28] J. Lapa, J. Bernardino, and A. Figueiredo, "A comparative analysis of open source business intelligence platforms," in *Proceedings of the International Conference on Information Systems and Design of Communication*, ISDOC '14, (New York, NY, USA), pp. 86–92, ACM, 2014.

[29] V. M. Parra, A. Syed, A. Mohammad, and M. N. Halgamuge, "Pentaho and Jaspersoft : A Comparative Study of Business Intelligence Open Source Tools Processing Big Data to Evaluate Performances," *IJACSA ) International Journal of Advanced Computer Science and Applications*, vol. 7, no. 10, 2016.

[30] J. B. António Marinheiro, "Analysis of open source business intelligence suites," in *Proceedings of the International Conference on Information Systems and Design of Communication*, (Lisboa, Portugal), IEEE, 2013.

[31] M. Casters, R. Bouman, and J. v. Dongen, *Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration.* Wiley Publishing, 2010.